ServiceOptions: Руководство программиста

DocumentId:GradSoft-PR-02.07.2000-v1.0.3

August 28, 2002

Contents

1 Введение

ServiceOptions представляет собой компоненту для обработки опций командной строки стандартного CORBA сервера.

Что это означает: обычно CORBA сервер экспортирует в "мир" несколько объектов: т. н. "Корневые Объекты" (Root Objects) сервиса. К примеру, CollectionService экспортирует **CollectionFactory** и **RACollectionFactory** ; NamingService экспортирует **RootNamingCotext** .

Пользователи сервиса получают доступ к этим объектам с помощью механизмов, специфичных для каждой ORB.

Для стандартных CORBA сервисов это вызов *ORB::resolve_initial_references(name)* ; но процесс задания инициальных ссылок все равно требует знания объектных ссылок во время старта клиента.

Обычно, "корневые объекты" сервисов экспортируются с помощью:

- IOR в стиле corbaloc.
- вывода объектных ссылок в файл.
- регистрации объекта в NamingService.
- регистрации объекта в репозитарии имплементаций.

Что делает Service Options: инкапсулирует всю эту работу и использование специфичных непортабельных механизмов ORB в одном вызове. Вы просто указываете в коде сервера сервант и его имя; после этого ваш сервант доступен для использования в corbaloc ior-ах и автоматически публикуется другими методами, в зависимости от опций командной строки.

ServiceOptions основан на ProgOptions (`www.gradsdoft.kiev.ua/rus/Products/ToolBox/ProgOptions/ProgGuide/ProgrammingGuide_rus.html`)

Этот документ представляет собой неформальное описание, полная спецификация пакета приводится в API reference (`www.gradsoft.kiev.ua/common/ToolBox/ServiceOptions/API/index.html`).

2 Общее описание

Программист:

1. Создает объект типа ServiceOptions.
2. Указывает имена поддерживаемых CORBA сервисов при помощи метода `ServiceOptions::putServiceName`
3. Вызывает метод `ServiceOptions::parse` - командная строка будет разобрана во внутренние структуры ServiceOptions, при этом будут автоматически обработаны опции `--help` и `--config <filename>` (детали смотри в руководстве пограммиста к пакету ProgOptions);
4. При старте сервера после создания корневого объекта сервиса вызвает метод `ServiceOptions::bindServiceObject`

Результатом этих действий будет способность программы воспринимать опции:

- `--with-naming` - инициальные объекты сервисов отображаются в Name-Service
- `--ior-stdout` - объектыне ссылки в виде строки печатаются на стандартном выводе программы.
- `--ior-file-<name>` - поместить объектную ссылку сервиса с именем `<name>` в файл аргумента этой опции.

в дополнение к стандартным опциям ORB

3 Простейший пример

иллюстрирующий использование пакета ServiceOption приведен ниже:

A. Пусть у нас имеется следующий IDL-интерфейс:

```
interface HelloWorlder
{
  void hello_world();
};
```

B. Тогда мы можем написать такую программу:

```
#include <tao/CORBA.h>                          //
#include <tao/PortableServer/PortableServer.h>  // Допустим, у нас TAO-1.2
#include <orbsvcs/CosNamingC.h>                 //
#include <HelloWorlderS.h>                      //

#include <iostream>
using namespace std;

#include <GradSoft/ServiceOptions.h>  // не забудь
using namespace GradSoft;             //

class HelloWorlder_impl:public POA_HelloWorlder           // реализуем
{                                                          // интерфейс
public:                                                    //
  void hello_world() { cout << "Hello, world" << endl; }  //
};                                                         //

int main(int argc, char** argv)
{
 ServiceOptions options;
 options.putServiceName("HelloService");   // установить имя поддерживаемого сервиса
 if (!options.parse(argc,argv)) return 1;  // разобрать опции

 // инициализация ORB: используем копию внутреннего
 // (объединенного) вектора аргументов ServiceOptions,
 // для того чтобы ORB_init() мог получить опции командной строки
 // в конфигурационном файле формата ProgOptions:

 ProgOptions::ArgsHolder argsHolder;
 argsHolder.takeArgv(options);
 CORBA::ORB_var orb = CORBA::ORB_init(argsHolder.argc,argsHolder.argv);

 CORBA::Object_var poaObj =                                    // стандартные
     orb->resolve_initial_references("RootPOA");               // действия
 PortableServer::POA_var poa = PortableServer::POA::_narrow(poaObj);  //
 PortableServer::POAManager_var poaManager = poa->the_POAManager();   //
```

```
 poaManager->activate();                                                    //

 HelloWorlder_impl helloWorlder_impl;   // создать объект

 HelloWorlder_var helloWorlder = helloWorlder_impl._this();

 options.bindServiceObject(     // главный вызов
    orb,
    helloWorlder,
    &helloWorlder_impl,
    "HelloService",
    true
 );

 orb->run();
 orb->destroy();
 return 0;
}
```

Эта программа будет обрабатывать:

1. опции `--with-naming`, `--ior-stdout`, `--ior-file-myName` так как это было описано выше;
2. опции `--help` и `--config` как было описано в Руководстве программиста к пакету ProgOptions;
3. все обычные опции ORB.

4 Требования к программному окружению

1. Должны выполняться требования к программному окружению пакета ProgOptions (см. ProgOptions Programming Guide)
2. Если Вы используете TAO-1.2, вы должны включить заголовочный файл PortableServer.h перед включением ServiceOptions.h

5 Перечень изменений

06.02.2002 – модифицирован пример использования: ORB_init получает копию "объединенного" вектора аргументов, полученную при помощи метода getArgsHolder()

17.01.2002 – модифицирован пример использования

03.01.2002 – коррекция под новый релиз GradC++ToolBox 1.4.0

18.02.2001 – просмотр, добавленна ссылка на corbaconf и формальные аттрибуты эксплуатационной документации.

02.07.2000 – первая редакция.