ServiceOptions: Programming Guide

DocumentId:GradSoft-PR-e-02.07.2000-v1.0.3

August 28, 2002

Contents

1 Introduction

ServiceOptions is a component for handling of typical command options of CORBA Service-like program.

What it mean: usual CORBA Service export to "world" few objects: so called "Root Objects" of the service. For example, Collecion Service exports **CollectionFactory** and **RACollectionFactory** ; NamingService exports **RootNamingContext**. Service users obtaine access to such services via propierty mechanizme of ORB.

For standart CORBA services this is call of *ORB::resolve_initial_references(name)*; but process of setting initial references in ORB with -ORBIntRef option require knowing IOR-s of using objects during start of the program.

Usually, "root objects" of the service are exported by

- Setting corbaloc-style IOR-s.
- Publishing objects IOR-s in file.
- Registering objects in naming service.
- Registering objects in IMR.

What do ServiceOptions: It's incapsulate all this work in one method call. I. e. you just point to serlet and name; after this you object is accessible via corbaloc-style IOR and optionally is published in other ways, depends from command line options of you server.

ServiceOptions is based on ProgOptions (`www.gradsdoft.kiev.ua/eng/Products/ToolBox/ProgOptions/ProgGuide/ProgrammingGuide_eng.html`)

This document is an unformal description, for full specification, please, use API reference (`http://www.gradsoft.kiev.ua/common/ToolBox/ServiceOptions/API/index.html`).

2 General Description

Programmer must perform the next steps:

1. Create object of type ServiceOptions.
2. Set names for provided CORBA objects by calling `ServiceOptions::putServiceName`
3. Call `ServiceOptions::parse` for parsing options (options `--help` and `--config <filename>`, if exist, will be handled, see ProgOptions ProgrammingGuide for detail)
4. After creating root objects call method `ServiceOptions::bindServiceObject`

In result, root object references will be accessible via corbaloc-style URL (corbaloc::host:port/Name) and program will understood next options:

- `--with-naming` - initial objects are registered in NameService
- `--ior-stdout` - stringifized object references are printed on standart output of a program.
- `--ior-file-<name>` - IOR of object wiht name `<name>` will be published to file with name `<argument of thid option>`

in addition to common ORB options.

3 Trivial example

Trivial example which illustrate ServiceOptions usage is follow:

A. Once the following IDL-interface exist:

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

B. Then we can write next programm:

```
#include <tao/CORBA.h>                            //
#include <tao/PortableServer/PortableServer.h>  // for TAO-1.2
#include <orbsvcs/CosNamingC.h>                   //
#include <HelloWorlderS.h>                        //

#include <iostream>
using namespace std;

#include <GradSoft/ServiceOptions.h>  // do'nt forget
using namespace GradSoft;             //

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

int main(int argc, char** argv)
{
 ServiceOptions options;
 options.putServiceName("HelloService");   // set service name
 if (!options.parse(argc,argv)) return 1;  // parse options set

// use copy of ServiceOptions's internal (argument count,argument vector) pair
// to make it possible
// ORB_init() takes comand-line options in ProgOptions config file:

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

 CORBA::Object_var poaObj =                                          // standard
     orb->resolve_initial_references("RootPOA");                     // steps
 PortableServer::POA_var poa = PortableServer::POA::_narrow(poaObj);    //
 PortableServer::POAManager_var poaManager = poa->the_POAManager();     //
 poaManager->activate();                                             //

 HelloWorlder_impl helloWorlder_impl;   // create object

 HelloWorlder_var helloWorlder = helloWorlder_impl._this();

 options.bindServiceObject(     // principal call
    orb,
```

```
    helloWorlder,
    &helloWorlder_impl,
    "HelloService",
    true
 );

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

This program handle next options:

1. `--with-naming`, `--ior-stdout`, `--ior-file-myName` as it was described above;
2. `--help --config` as it was described in Programming guide for ProgOptions package;
3. current ORB options.

4 Programming Environment Conventions

1. ProgOptions Programming Environment Conventions must be set. (see ProgOptions Programming Guide)
2. Using ServiceOptions with TAO-1.2, you must to include header file PortableServer.h before including of ServiceOptions.h.

5 Changes

06.02.2002 - example modified: run ORB_init with copy of argument vector obtained by using getArgsHolder() is accepted

17.01.2002 - example verified

03.01.2002 - touch before 1.4.0 public release.

18.02.2001 - review, added formal document attributes.

02.07.2000 - initial revision.