

Gem #111: Приложение к стандарту языка «Распределённые системы» (PPC) (The Distributed Systems Annex) (DSA)

Часть 5 - Встроенный блок преобразования имён

Автор: Thomas Quinot, AdaCore

Краткое содержание: Этот Gem - пятый в серии публикаций, показывающих возможности вспомогательного приложения по распределённым системам, описанного в Справочном Руководстве Ada (Приложение E). В предыдущем Gem (#90) было продемонстрировано, как осуществлять сопряжение кода DSA с приложением C/C++ в виде автономной библиотеки Ada.

В Gem #111 мы покажем, как блок преобразования имён DSA можно встроить в главную подобласть определения вместо того, чтобы запускать его как автономный процесс.

Давайте начнём...

В первом выпуске этой серии DSA Gems мы использовали приложение, управляющее публичной доской объявлений, как хороший пример дизайна клиент-сервер. Мы использовали следующую конфигурацию:

```
configuration Dist_App is
  pragma Starter (None);
  -- User starts each partition manually
  ServerP : Partition := (Bulletin_Board);
  -- RCI package Bulletin_Board is on partition ServerP
  ClientP : Partition := ();
  -- Partition ClientP has no RCI packages
  for ClientP'Termination use Local_Termination;
  -- No global termination
  procedure Display_Messages is in ServerP;
  -- Main subprogram of master partition
  procedure Post_Message;
  for ClientP'Main use Post_Message;
  -- Main subprogram of slave partition
end Dist_App;
```

и мы получили два исполняемых файла: по одному для каждого раздела (serverp и clientp), выпустив следующую команду:

```
po_gnatdist dist_app
```

При вышеуказанной конфигурации po_gnatdist запуск приложения требует трех шагов:

1. Запустите po_cos_naming, сервер имен PolyORB. При запуске отображается ссылка на объект, которая должна быть передана всем разделам в переменной среды POLYORB_DSA_NAME_SERVICE или через файл конфигурации PolyORB.
2. Запустите serverp, раздел сервера, который регистрирует свой RCI-пакет (Bulletin_Board) с сервером имен.
3. Запустите clientp, клиентский раздел, который будет запрашивать сервер имён для местоположения RCI.

Упрощение процесса

Чтобы упростить весь этот процесс, вы можете вместо этого установить `po_gnatdist` для встраивания сервера имён в основной раздел (`serverp`). Это достигается путём:

```
pragma Name_Server (Embedded);
```

в файле конфигурации.

Затем вы можете запустить `serverp` без внешнего сервера имён: теперь он включён в раздел. Вам все равно необходимо передать местоположение сервера имён клиентам. Изнутри раздела сервера эту информацию можно получить, запросив параметры времени выполнения `PolyORB`:

```
Put_Line ("ServerP started, embedded name server is at:");  
Put_Line (PolyORB.Parameters.Get_Conf ("dsa", "name_service", ""));
```

Это выводит строку формы:

```
IOR:<...long series of hex digits>
```

который кодирует всю необходимую информацию. Затем вы можете запустить клиентский раздел, указав это значение в переменной среды `POLYORB_DSA_NAME_SERVICE`. В синтаксисе оболочки Bourne это означает:

```
POLYORB_DSA_NAME_SERVICE=IOR:<...> ./clientp
```

Используя Windows cmd оболочку, это было бы:

```
set POLYORB_DSA_NAME_SERVICE=IOR:<...>  
client
```

Передача информации о сервере имён в файл

`POLYORB_DSA_NAME_SERVICE` также может указывать имя файла с префиксом «file:». Итак, если вы измените `serverp` для вывода ссылки на службу имён в файл «ns.txt», вы можете запустить клиент, используя

```
POLYORB_NAME_SERVICE=file:ns.txt ./clientp
```

Используя закрепленный порт

Note: this requires the latest development version of PolyORB.

Примечание: для этого требуется последняя версия разработки PolyORB.

Иногда неудобно переносить строковое значение или файл с сервера на клиент и обновлять его при каждом перезапуске сервера. Используя соответствующие директивы конфигурации среды выполнения `PolyORB`, можно заставить сервер прослушивать сетевые подключения в фиксированном расположении.

Следующая конфигурация заставляет сервер слушать порт 8889:

```
[iiop]  
polyorb.protocols.iiop.default_port=8889
```

Как только это установлено для сервера, Вы можете направить клиент к тому расположению установкой `POLYORB_DSA_NAME_SERVICE` к:

```
corbaloc:iiop:1.2@:8889/_NameService
```

Включённые в файлы start_server и start_client сценарии предоставляют демонстрацию этого средства.

Связанный со статьёй текст программы

Attached Files [05-embedded-ns.zip](#)

Файл start_client:

```
#!/bin/sh

if [ $# != 2 ]; then
  echo "Usage: $0 <host> <port>"
  exit 1
fi

POLYORB_DSA_NAME_SERVICE=corbaloc:iiop:1.2@$1:$2/_NameService \
./client
```

Файл start_server:

```
#!/bin/sh

if [ $# != 1 ]; then
  echo "Usage: $0 <port>"
  exit 1
fi

POLYORB_IIOP_POLYORB_PROTOCOLS_IIOP_DEFAULT_PORT=$1 \
./serverp
```

Файл dist_app.cfg:

```
configuration Dist_App is
  pragma Starter (None);
  -- User starts each partition manually

  pragma Name_Server (Embedded);
  -- Embed NS in main partition

  ServerP : Partition := (Bulletin_Board);
  -- RCI package Bulletin_Board is on partition ServerP

  ClientP : Partition := ();
  -- Partition ClientP has no RCI packages

  for ClientP'Termination use Local_Termination;
  -- No global termination

  procedure Display_Messages is in ServerP;
  -- Main subprogram of master partition

  procedure Post_Message;
  for ClientP'Main use Post_Message;
  -- Main subprogram of slave partition
end Dist_App;
```

Файл bulletin_board.ads:

```
1. package Bulletin_Board is
2.   pragma Remote_Call_Interface;
3.--   This makes the package a Remote Call Interface (RCI), so the subprograms
4.   -- below are remotely callable.
5.
6.   -- This pragma enforces some restrictions on the unit to ensure that any
7.   -- visible subprogram can actually be called remotely, and in particular
8.   -- that the types of the parameters are suitable for transport over a
9.   -- communication link from one partition to another.
10.
11.  subtype Length is Natural range 0 .. 100;
12.  type News_Item (Author_Length, Message_Length : Length := 0) is record
13.    Author   : String (1 .. Author_Length);
14.    Message  : String (1 .. Message_Length);
15.  end record;
16.
17.  type News_Items is array (Positive range <>) of News_Item;
18.
19.  procedure Post (Item : News_Item);
20.  function Whats_Up return News_Items;
21. end Bulletin_Board;
```

Файл bulletin_board.adb:

```
1. package body Bulletin_Board is
2.   Store : News_Items (1 .. 64);
3.   Last  : Natural := Store'First - 1;
4.
5.   procedure Post (Item : News_Item) is
6.   begin
7.     Last := Last + 1;
8.     Store (Last) := Item;
9.   end Post;
10.
11.  function Whats_Up return News_Items is
12.  begin
13.    return Store (Store'First .. Last);
14.  end Whats_Up;
15.
16. end Bulletin_Board;
```

Файл display_messages.adb:

```
1. with Ada.Text_IO; use Ada.Text_IO;
2. with Bulletin_Board; use Bulletin_Board;
3. with PolyORB.Parameters;
4. procedure Display_Messages is
5. begin
6.   Put_Line ("ServerP started, embedded name server is at:");
7.   Put_Line (PolyORB.Parameters.Get_Conf ("dsa", "name_service", ""));
8.
9.   loop
10.    Put_Line ("----- all messages -----");
11.    declare
12.      Contents : constant News_Items := Whats_Up;
```

```

13.     begin
14.         for J in Contents'Range loop
15.             Put_Line (Contents (J).Author & " says:");
16.             Put_Line (Contents (J).Message);
17.             New_Line;
18.         end loop;
19.         delay 2.0;
20.     end;
21. end loop;
22. end Display_Messages;

```

Файл `post_message.adb`:

```

1. with Ada.Text_IO; use Ada.Text_IO;
2. with Bulletin_Board; use Bulletin_Board;
3. procedure Post_Message is
4.     Author, Message : String (1 .. 140);
5.     Author_Length, Message_Length : Natural;
6. begin
7.     Put ("Author name: ");
8.     Get_Line (Author, Author_Length);
9.     Put ("Message : ");
10.    Get_Line (Message, Message_Length);
11.
12.    Post (News_Item'
13.        (Author_Length => Author_Length,
14.         Message_Length => Message_Length,
15.         Author         => Author (1 .. Author_Length),
16.         Message        => Message (1 .. Message_Length)));
17.    -- This subprogram call may be remote, but we write it exactly in the
18.    -- usual way.
19. end Post_Message;

```

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Thomas Quinot (Томас Аквинский) имеет степень инженера от Télécom Paris и доктора философии от Université Paris VI. Основной вклад его исследовательской работы - определение гибкой архитектуры промежуточного (middleware) программного обеспечения, направленной на взаимодействие модели распределения параллельных процессов с целью достижения функциональной совместимости. Он присоединился к AdaCore как Главный Разработчик программного обеспечения (a Senior Software Engineer) в 2003 и ответственен за технологии распределения. Он также участвует в разработке, обслуживании и поддержке компилятора GNAT.

Last Updated: 11/24/2017

Posted on: 10/3/2011

Обсуждение...