Ada 95 — programowanie rozproszone w przykładach

Dariusz Wawrzyniak
Dariusz.Wawrzyniak@cs.put.poznan.pl

Plan

Plan

Funkcje i procedury zdalne

- Procedury i funkcje zdalne w języku Ada 95 tworzy się poprzez zadeklarowanie ich w jednostce bibliotecznej kategorii `Remote_Call_Interface`.
- Procedury te udostępniane są i tym samym wykonywane przez partycję aktywną, w skład której wchodzi dana jednostka biblioteczna.
- Z perspektywy modelu klinet-serwer partycja taka jest serwerem procedur zdalnych, zadeklarowanych we wchodzących w jej skład jednostkach bibliotecznych.
- Kategoryzacja jednostki bibliotecznej odbywa się poprzez umieszczenie odpowiedniej pragmy (w przypadku procedur zdalnych jest to `pragma Remote_Call_Interface`).

Specyfikacja pakietu procedur zdalnych

```
package Server is
   pragma Remote_Call_Interface;

   procedure Service(arg: in out Positive);
end Server;
```

Przykład wywołanie procedury zdalnej

```
with Server;
with Text_IO; use Text_IO;

Procedure main is
   value: Positive := 34;
begin
   Server.Service(value);
   Put_line("Result obtained:" & Integer'Image(
      value));
end main;
```

Przykład pliku konfiguracyjnego

```
configuration Messenger is

   -- pragma Starter (None);
   -- uruchamianie reczne

   pragma Boot_Server ("tcp", "localhost:5556");

   Msg_Client: Partition;
   Msg_Server: Partition := (Server);
   Procedure main is in Msg_Client;
end Messenger;
```

Plan

Procedury asynchroniczne

- Procedurę asynchroniczną towrzy się z użyciem pragmy `Asynchronous`.
- Asynchorniczne procedury zdalne nie mogą mieć parametryów typu *out* i *in out*.
- Nie można również definiować funkcji asynchronicznych.
- Wyjątki zgłaszane w czasie wykonania procedury asynchronicznej, które nie zostaną obsłużone w tej procedurze są ignorowane.

Specyfikacja asynchronicznej procedury zdalnej

```
package Server is
   pragma Remote_Call_Interface;

   procedure Service(arg: Positive);
   function Get_result return Positive;
   pragma Asynchronous(Service);
end Server;
```

Przykazywanie wyników wywołania asynchronicznego

- Przechowanie wyniku po stronie serwera do czasu odebrania go przez klienta. Odebranie może nastąpić w wyniku wykonania innej procedury lub funkcji.
  - Jak długo przechowywać wynik?
  - W jaki sposób informować klienta o gotowości wyniku?
  - Co zrobić, gdy przed pobraniem wyniku pojawi się następne wywołanie asynchroniczne?
  - W jaki sposób kojarzyć klienta z oczekującym na niego wynikiem w przypadku współbieżnych wywołań?
- Przekazanie wyniku poprzez wywołanie zwrotne po zakończeniu wykonywania procedury właściwej zdalnej.
  - Czy można po stronie klienta udostępnić procedurę zdalną wywoływaną zwrotnie przez serwer?
  - W jaki sposób przekazać serwerowi informację o procedurze zwrotnej (jej identyfikator)?

Udostępnianie wyników wywołania asynchronicznego

```
package body Server is
   result: Positive;

   procedure Service(arg: Positive) is
   begin
      delay 5.0;
      result := arg + 10;
   end Service;

   function Get_result return Positive is
   begin
      return result;
   end;
end Server;
```

Wywołanie procedury asynchronicznej

```
with Server;
with Text_IO; use Text_IO;

Procedure main is
begin
   Server.Service(34);
   delay 7.0;
   Put_line("Result obtained:" & Integer'Image(
      Server.Get_result));

   Server.Service(20);
   Put_line("Result obtained:" & Integer'Image(
      Server.Get_result));
   delay 7.0;
   Put_line("Result obtained:" & Integer'Image(
      Server.Get_result));
end main;
```

Plan

Plan

Koncepcja zwrotnego wywołania zdalnej procedury

- Po stronie klienta udostępniany jest pakiet procedur zdalnych.
- Klient przekazuje serwerowi wskaźnik (zdalny, fat pointer) na procedurę zdalną, którą udostępnia w celu przekazania wyniku.
- Serwer wywołuje wskazaną procedurę w celu przekazania wyniku.

Wywołanie zwrotne (klient)

```
package Client is
   pragma Remote_Call_Interface;

   procedure Put_result(arg: Positive);
   procedure Remote_call_wrapper(arg: Positive);
end Client;
```

Wywołanie zwrotne (serwer)

```
package Server is
   pragma Remote_Call_Interface;

   type RR_reference is access procedure (arg:
      Positive);

   procedure Service(arg: in Positive; rrf: in
      RR_reference);
   pragma Asynchronous(Service);
end Server;
```

Przykład implementacji pakietu serwera

```
package body Server is
   procedure Service(arg: in Positive; rrf: in
       RR_reference) is
   begin
      delay 5.0;
      rrf.all(arg+10);
   end Service;
end Server;
```

Przykład implementacji pakietu klienta

```
with Server;
with Text_IO; Use Text_IO;

package body Client is
   procedure Put_result(arg: Positive) is
   begin
      Put_line("The result is "& Positive'Image(
         arg));
   end Put_result;

   procedure Remote_call_wrapper(arg: Positive)
      is
   begin
      Server.Service(arg, Put_result'Access);
   end Remote_call_wrapper;
end Client;
```

Przykład programu klienta

```
with Server;
with Client;
with Text_IO; use Text_IO;

Procedure main is
begin
   Client.Remote_call_wrapper(34);
   for i in 1..10 loop
      Put_line("Waiting for the result");
      delay 1.0;
   end loop;
   Put_line("Result obtained");
end main;
```

Synchronizacja klienta z wywołaniem zwrotnym

```
with Server;
with Text_IO; Use Text_IO;

package body Client is
   protected Result_buffer is
      procedure put(v: in Positive);
      entry get(v: out Positive);
      private
         flag: Boolean := False;
         value: Positive;
   end Result_buffer;

   procedure Put_result(arg: Positive) is
   begin
      Result_buffer.put(arg);
   end Put_result;
```

Plan

Ograniczenia w zastosowaniu procedur zdalnych

- Pakiet procedur musi być unikalny w aplikacji rozproszonej, tzn. ten sam pakiet nie może występować w kilku różnych partycjach.
- Program danej partycji może zostać uruchomiony tylko 1 raz w jednym z węzłów.
- Jeśli określona procedura miałaby być wywoływana zwrotnie w kilku różnych partycjach musiałaby być definiowana w oddzielnym dla każdej partycji pakiecie o unikalnej nazwie.

Przykład wielokrotnego wystąpienia pakietu w partycjach

```
configuration Messenger is
   pragma Starter (None);

   pragma Boot_Location ("tcp", "localhost:5556");

   Msg_Client: Partition := (Client);
   Msg_Client1: Partition := (Client);
   Msg_Client2: Partition := (Client);
   Msg_Server: Partition := (Server);

   Procedure Start is in Msg_Server;
   Procedure main;
   for Msg_Client'Main use main;

   for Msg_Server'Termination use Local_Termination
      ;
```

Koncepcja zwrotnego wywołania zdalnej metody

- Po stronie klienta udostępniany jest pakiet definiujący zdalny typ znakowany.
- Klient przekazuje serwerowi wskaźnik (zdalny, fat pointer) na obiekt zdalnego typu znakowanego, dla którego zdefiniowana jest określona metoda na potrzeby przekazania wyniku.
- Serwer wywołuje metodę (procedurę) z przekazanym przez klienta wskaźnikiem jako pierwszym parametrem oraz wynikiem procedury zdalnej jako drugiem parametrem. W ten sposób procedura wykona się po stronie klienta i umożliwi przekazanie wyniku.

Specyfikacja zdalnego typu

```
package CallBack is
pragma Remote_Types;
   type CB_Class is tagged limited private;
   type CB_Pointer is access all CB_Class'Class;
   procedure Put_result(ptr: access CB_Class;
      arg: Positive);
private
   type CB_Class is tagged limited null record;
end CallBack;
```

Przykład implementacji zdalnej metody

```
with Text_IO; use Text_IO;

package body CallBack is
   procedure Put_result(ptr: access CB_Class;
      arg: Positive) is
   begin
      Put_Line("The result is " & Positive'Image
         (arg));
   end Put_result;
end CallBack;
```

Wywołanie zwrotne poprzez zdalny obiekt (klient)

```
package Client is
   procedure Remote_call_wrapper(arg: Positive);
end Client;
```

Wywołanie zwrotne poprzez zdalny obiekt (serwer)

```
with CallBack; use CallBack;

package Server is
pragma Remote_Call_Interface;

   --type RR_reference is access all CB_Class'
      Class;

   procedure Service(arg: in Positive; rrf:
      CB_Pointer);
pragma Asynchronous(Service);
end Server;
```

Przykład implementacji pakietu serwera

```
package body Server is
   s: Positive := 1;
   procedure Service(arg: in Positive;
                     rrf: in CB_Pointer) is
   begin
      delay 5.0;
      Put_result(rrf, s);
      s := arg;
   end Service;
end Server;
```

Przykład implementacji pakietu klienta

```
with Server;
with Text_IO; use Text_IO;
with CallBack; use CallBack;

package body Client is
   obj: aliased CB_Class;
   procedure Remote_call_wrapper(arg: Positive)
      is
   begin
      Server.Service(arg, obj'Access);
   end Remote_call_wrapper;
end Client;
```