Gem #117: Проектировочный шаблон: Переопределяемые атрибуты класса в Ada 2012

Автор: Emmanuel Briot, AdaCore

Краткое содержание: В данном Gem будет обсуждаться реализация «атрибутов класса» (таким же образом, как в языке Python) в Ada.

Давайте начнём…

Большинство объектно-ориентированных языков программирования предоставляют возможность для объявления переменных, которые совместно используются всеми объектами данного класса. В C ++ они называются «статическими членами» “`static members`” (и используют ключевое слово «`static`»), и аналогично Python имеет понятие «атрибуты класса».

Рассмотрим пример, где это полезно. Например, предположим, что мы хотим определить понятие блока текста, который генерируется путём расширения шаблона (возможно, после замены некоторых параметров в этом шаблоне, как это может быть сделано с помощью анализатора шаблонов AWS, например). Как только мы вычислили эти параметры, нам может понадобиться генерировать несколько выходов (например, HTML и CSV). Только шаблон необходимо изменить, а не вычислять параметры.

Как правило, например, в Python, шаблон может быть реализован как атрибут класса, в нашем примере класса `Text_Block`. Затем мы можем создавать шаблоны, которые нуждаются в одной и той же информации, но имеют другой выход, просто расширяя этот класс:

```
class Text_Block(object):
    template = "somefile.txt"
    def render (self):
        # ... compute some parameters
        # Then do template expansion
        print "processing %s" % self.__class__.template

class Html_Block(Text_Block):
    template = "otherfile.html"
```

В этом примере мы решили использовать атрибут класса, а не обычный атрибут экземпляра (`self.template`). Этот пример исходит из реализации `GnatTracker`: на веб-сервере мы создаём новый экземпляр `Text_Block` для каждого запроса, который мы должны обслуживать. Для этого мы используем реестр, который сопоставляет URL-адрес класса, который нам нужно создать. Таким образом, проще создать новый экземпляр без указания имени шаблона в качестве параметра, который потребуется, если имя шаблона было сохранено в экземпляре. Ещё одна причина (хотя и не совсем применимая здесь) - сохранить память, что было бы важно в тех случаях, когда тысячи экземпляров класса.

Разумеется, подход, предложенный в этом Gem, - это не единственный способ решить основную проблему, но он служит хорошим примером одной из новых функций Ada 2012.

C ++, как и Ada, не даёт возможности переопределить статический член класса, поэтому он будет использовать аналогичное решение, как описано ниже.

Поскольку Ada не имеет понятия об атрибуте `overridable class`, мы будем его моделировать, используя вместо него подпрограмму (единственный способ получить диспетчеризацию в Ada). Важным моментом здесь является то, что мы хотим переопределить имя шаблона в дочерних классах, поэтому мы не можем использовать простую константу в спецификации пакета или теле.

```
type Text_Block is tagged null record;
function Template (Self : Text_Block) return String;
function Render (Self : Text_Block) return String;

function Template (Self : Text_Block) return String is
   pragma Unreferenced (Self);
begin
   return "file_name.txt";
end Template;
```

Параметр `Self` используется только для диспетчеризации (чтобы потомки `Text_Block` могли переопределить эту функцию). Поскольку мы предпочитаем компилировать с `"- gnatwa"`, чтобы получить предупреждение о неиспользуемых сущностях, мы указываем компилятору, что ожидается, что `Self` не имеет ссылки.

Мы могли бы сделать шаблон функции `inlinable`, что может быть полезно в некоторых случаях (например, при вызове из `Render` в вызове `nondispatching`), но в целом не будет никакой пользы, потому что `Template` будет диспетчерским вызовом, который требует косвенного вызова и, таким образом, не выиграет от встраивания.

И на этом все. У нас есть Ada-эквивалент члена класса Python.

Но пока ничего нового здесь нет, и этот подход довольно тяжёлый для написания. Например, тело `Render` может содержать код вида:

```
pragma Ada95;

function Render (Self : Text_Block) return String is
   T : constant String := Template (Text_Block'Class (Self));
begin
   ..  prepare the parameters for template expansion
   ..  substitute in the template and return it
end Render;
```

К счастью, Ada 2012 предоставляет более простой способ написать это, используя новые возможности синтаксиса – функцию выражения. Поскольку `Template` – это функция, которая возвращает константу, мы можем объявить её непосредственно в спецификации и полностью удалить тело. Таким образом, Спецификация будет выглядеть так:

```
pragma Ada_2012;

type Text_Block is tagged null record;
function Template (Self : Text_Block) return String
   is ("filename.txt");
function Render (Self : Text_Block) return String;
```

Это гораздо более лёгкий синтаксис и гораздо ближе к тому, как это можно сделать в Python (за исключением того, что мы используем функцию вместо переменной для представления члена класса). Потомок `Text_Block` будет переопределять шаблон с использованием тех же обозначений:

```
type Html_Block is new Text_Block with null record;
overriding function Template (Self : Text_Block) return String
   is ("otherfile.html");
```

По сравнению с Python это на самом деле более мощно, потому что некоторые из них могут предоставить более сложное тело для шаблона, поэтому мы не ограничиваемся использованием значения простой переменной, как в Python. Фактически, мы можем сделать это в самой спецификации, используя условное выражение (ещё одна новая особенность Ada 2012):

```
pragma Ada_2012;

type Text_Block is tagged null record;
function Template (Self : Text_Block) return String
    is (if Self.Blah then "filename.html" else "file2.json");
function Render (Self : Text_Block) return String;
```

Наконец, мы также можем сделать тело `Render` немного более знакомым (с точки зрения объектно-ориентированной нотации) с использованием точечной нотации, введённой в Ada 2005:

```
function Render (Self : Text_Block) return String is
   T : constant String := Text_Block'Class (Self).Template;
begin
   ..  prepare the parameters for template expansion
   ..  substitute in the template and return it
end Render;
```

Теперь вызов `Template` выглядит ближе к тому, как он будет выглядеть в тех языках, которые предоставляют переопределяемые члены класса. Некоторые будут утверждать, что это не похоже на вызов функции и, следовательно, менее читаемо, так как мы не знаем, что мы вызываем функцию. Это дело вкуса, но, по крайней мере, у нас есть выбор.

Есть одна вещь, которую мы временно потеряли в объявлении шаблона. Если мы компилируем с `-gnatwa`, компилятор будет жаловаться, что `Self` не имеет ссылок. В настоящее время невозможно добавить директиву `pragma` без ссылки в функцию выражения. Это породило дискуссии в AdaCore и вопрос ещё не решён. В настоящее время два предложения - либо всегда опускать предупреждение неиспользуемого параметра, когда функция имеет один параметр и управляет диспетчеризацией (именно для облегчения этого шаблона члена класса), либо использовать аспект Ada 2012 для этого, как показано ниже:

```
function Template (Self : Text_Block) return String
   is ("filename.html")
with Unreferenced => Self;
```

Также обратите внимание, что использование функций выражения в этом Gem требует самой последней версии GNAT: функция выражений не была доступна в более старых версиях, и первоначальная реализация имела некоторые ограничения.

Связанный со статьёй текст программы

Attached Files отсутствуют

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

Об авторе

Emmanuel Briot
AdaCore

Emmanuel Briot был с AdaCore с 1998 года. Он участвовал в различных проектах, в частности, ориентированных на графические пользовательские интерфейсы, включая GtkAda, GPS, XML / Ada,

GnatTracker и наш внутренний CRM. Он имеет инженерную степень в Ecole Nationale des Telecommunications (Брест, Франция).

Last Updated: 10/13/2017
Posted on: 2/1/2012

Обсуждение…

Один ответ на «Gem # 117: Design Pattern: Overridable Class Attributes в Ada 2012»

1. 6 февраля 2012 года в 20:59

Christoph Grein сказал:

Есть небольшая опечатка, я уверен, что это то, что вы имеете в виду:

```
type Html_Block is new Text_Block with null record;
overriding function Template (Self : HTML_Block) return String
is (“otherfile.html”);
```

Для подавления предупреждения я проголосовал бы за конкретный вариант реализации (если им разрешено, не знаю, слишком ленив, чтобы запросить RM):

```
with Unreferenced => Self;
```

Это может быть добавлением для следующего пересмотра Ada.