Gem #141: Разберитесь с конфигурацией

Автор: Emmanuel Briot — AdaCore

Краткое содержание: В серии Gem, посвященной `GNAT.Command_Line` (#138 и #139), мы упомянули, что есть несколько способов осуществления контроля над поведением приложения со стороны пользователя - Операции с командной строкой (описанные в данных Gem), графические пользовательские приложения (например, `GtkAda`) и файлы конфигурации. Именно о файлах конфигурации и пойдет речь в данной публикации

Давайте начнём…

В серии Gem на `GNAT.Command_Line` (Gems # 138 и # 139) мы упоминали, что есть несколько способов, которыми пользователь может контролировать поведение приложения. Это параметры командной строки (как обсуждалось в тех Gem), графические пользовательские приложения (например, использующие `GtkAda`) и файлы конфигурации. Этот Gem предлагает различные подходы для создание файлов конфигурации.

Файлы конфигурации - это файлы, редактируемые пользователем или машиной. Однако это общий термин, который не подразумевает какого-либо конкретного формата. Есть несколько форматов, которые часто используются, и мы обсуждаем несколько таких форматов в этом Gem.

XML файлы

XML очень часто используется для хранения данных, редактируемых машиной, но это не совсем удобный для человека формат и чрезвычайно подробный. Однако, среди его различные плюсы мощный набор инструментов для просмотра и редактирования XML-файлов, а также стандартные API для анализа и проверки таких файлов.

XML-файлы могут содержать любые типы данных (включая двоичные данные, хотя в этом случае они не являются лучшим подходом), и их можно проверить с помощью грамматик (также известных как XML-схемы).

Такие файлы могут быть проанализированы и проверены через библиотеку XML/Ada, как описано в Gem #21.

JSON файлы

JSON (JavaScript Object Notation)-это более поздний формат, который начал прочно закрепляться в современных приложениях, особенно в веб-ориентированных. Этот формат намного легче, и похож на JavaScript. Вот краткий пример такого файла:

```
{"configuration": {
    "key1": "value1",
    "key2": 12,
    "key3": [1, 2, 3]
}
```

Этот формат легко читается и доступен для записи. Одним из его недостатков является отсутствие синтаксиса для комментариев, которые не поддерживаются. Таким образом, документация должна быть помещена в отдельный файл. Однако существует множество библиотек (на нескольких языках программирования), которые можно использовать для анализа этих файлов. Прежде всего, все современные веб-браузеры полностью способны принимать и анализировать эти файлы, поэтому они являются удобным способом обмена данными (включая конфигурации, например) между веб-сервером и веб-клиентом.

В Ada вы можете использовать пакет GNATCOLL.JSON, часть коллекции компонентов GNAT. Будущий Gem опишет этот пакет более подробно, хотя пока вы можете посмотреть спецификации.

INI файл

Эти файлы были стандартными для приложений Windows и использовали.расширением ini, отсюда и их название. Этот формат файла включает в себя расплывчатый синтаксис, детали которого варьируются от одного приложения к другому. Вот пример такого файла:

```
[General]
# Some documentation
key1=value1
key2=12
key3=1
key3=2

[Section2]
filename=$HOME/value2
```

Этот формат содержит пары ключ-значение, опционально организованные в разделы. Комментарии могут быть вставлены, как показано в примере выше.

Такие файлы могут быть проанализированы пакетом GNATCOLL.Config, который также является частью коллекции компонентов GNAT.

Основное использование этого пакета заключается в следующем.

```
pragma Ada_05;
with GNATCOLL.Config;   use GNATCOLL.Config;

declare
   C : INI_Parser;
begin
   Open (C, "settings.txt");
   while not C.At_End loop
      Put_Line (C.Key & " = " & C.Value & " in " & C.Section);
      C.Next;
   end loop;
end;
```

Цикл повторяется по всем парам ключ-значение в файле, и для каждой возвращается значение в виде строки, целого числа или файла. Файлы могут быть указаны в файле конфигурации как абсолютные или относительные относительно местоположения файла, и GNATCOLL.Config автоматически нормализует их. В особом случае строка «$HOME» автоматически заменяется на местоположение домашнего каталога пользователя.

Поскольку грамматика не связана с файлом, приложение должно запросить значение в допустимом формате. Фактически, в приведенном выше примере значение «filename» может быть запрошено как строка или файл, в зависимости от контекста, хотя, конечно, оно не может быть запрошено как целое число.

В приведенном выше примере используется нотация «объектно-точечный» (префиксная форма вызовов подпрограмм), введенная в Ada 2005, но сама библиотека может использоваться с приложениями, не запрограммированными в Ada 2005.

Хотя этот API работает должным образом, пользователь должен где-то хранить значения, если он намеревается использовать их позже. GNATCOLL.Config также предоставляет пул для их хранения на время работы приложения. Это особенно удобно для предпочтений. Фактически, этот пул может использоваться для других форматов файлов конфигурации, как описано выше.

Можно проанализировать несколько файлов конфигурации. Обычно приложение анализирует общесистемный файл настроек, который содержит значения по умолчанию. Эти значения по умолчанию могут быть переопределены с помощью пользовательского файла настроек.

Пул автоматически запоминает, откуда было прочитано значение, чтобы он мог нормализовать имена файлов относительно данного файла настроек.

Код будет выглядеть так:

```
declare
   Pool   : Config_Pool;
   Parser : INI_Parser;
begin
   Open (Parser, "/system/settings.txt");
   Fill (Pool, Parser);
   Open (Parser, "/user/settings.txt");
   Fill (Pool, Parser);  --  override with user-specific settings

   Put_Line ("key1 is " & Pool.Get ("key1", Section => "General"));
   Put_Line ("key3 is "
             & Pool.Get ("key3", Section => "General", Index => 1)
             & ", "

             & Pool.Get ("key3", Section => "General", Index => 2));
   Put_Line ("filename is "
         & Pool.Get_File ("filename", "General").Display_Full_Name);
end;
```

Тем не менее, остается еще одна проблема с этим кодом. Если организация файла настроек изменяется (например, из-за того, что вы переименовываете ключ или перемещаете его в другой раздел), вам нужно будет найти все места в вашем коде, которые ссылаются на него, и изменить имя и / или раздел.

Поэтому, GNATCOLL.Config обеспечивает еще один подход, который должен определить ключи как глобальные переменные..

```
declare
   Filename : constant Config_Key := Create ("filename", "General");
   Key1     : constant Config_Key := Create ("key1", "General");
   Key3     : constant Config_Key := Create ("key3", "General");
begin
   Put_Line ("key1 is " & Key1.Get (Pool));
   Put_Line ("key3 is "
             & Key3.Get (Pool, 1) & ',' & Key3.Get (Pool, 2));
   Put_Line ("filename is " & Filename.Get_File.Display_Full_Name);
end;
```

Теперь, только единственное местоположение в коде должно быть изменено, когда формат конфигурационного файла изменяется.

По замыслу, GNATCOLL.Config можно расширить, объявив новые типы синтаксического анализатора, производные от Config_Parser. Эти анализаторы могут обрабатывать форматы XML или JSON, и они по-прежнему будут совместимы с описанным выше Config_Pool, что упрощает доступ к файлам конфигурации во всем приложении.

Связанный со статьёй текст программы

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

Об авторе

Автор: Emmanuel Briot - AdaCore

Эммануэль Брио работает в AdaCore с 1998 года. Он принимал участие в различных проектах, в частности, ориентированных на графические пользовательские интерфейсы, включая GtkAda, GPS, XML / Ada, GnatTracker и нашу внутреннюю CRM. Он получил степень инженера в Национальной школе телекоммуникаций - Брест, Франция (Ecole Nationale des Telecommunications - Brest, France).

Last Updated: 10/13/2017
Posted on: 2/11/2013

Обсуждение…