Gem #55: Введение в сопряжение Ada/Java

Автор: Quentin Ochem, AdaCore

Краткое содержание: Gem Ada #55 - Данный Gem будет последним в 2008 г. (время летит!). Новые Gem начнут публиковаться с 12 января 2009 г. А пока что наилучшие пожелания и весёлых праздников всем читателям!

Сопряжение Ada и Java - весьма сложная задача. В отличие от C, C++ или Fortran, Ada и Java работают в двух разных средах: Java - на JVM, Ada - напрямую на ОС. Поэтому невозможно напрямую связать функции Java и скомпилированный код на языке Ada с помощью прагмы Import. Разработчикам предлагаются два возможных решения данной проблемы: компилировать код напрямую в байт-код Java, используя GNAT с JVM, либо использовать Java Native Interface (JNI), с помощью которого можно осуществлять взаимодействие среды машинного кода и JVM. В данном Gem будет обсуждается второй подход.

Вручную использовать слой JNI довольно сложно и может привести к множеству ошибок. К счастью, в AdaCore доступен набор инструментов для автоматизации генерации интерфейсов с помощью GNAT-AJIS. Данный Gem - первый в серии публикаций, в которых будет продемонстрировано, как использовать этот набор инструментов чтобы создавать смешанные приложения языков Ada и Java.

Давайте начнем…

Предположим, у нас есть API (Application Program Interface) на стороне Ada, который мы хотели бы вызывать на Java. Этот API может основываться на наборе типов и функций, например:

```
package API is

   type R is record
      F1, F2 : Integer;
   end record;

   procedure Print (Str : String; V : R);

end API;
```

Наша цель - создать объект типа R из Java, а затем передать этот объект в процедуру Ada `Print` вместе со строковым параметром.

Генерация обязательного кода

Теперь мы должны генерировать код Java, а также дополнительный уровень JNI, который соединится с кодом Ada. Это может быть сделано, используя консольную утилиту `ada2java` из инструментария GNAT-AJIS, которую вызываем следующим образом:

```
ada2java api.ads –b test –o ada –c java –L my_lib
```

Поскольку инструмент только генерирует привязку к пакету, ему нужен только доступ к спецификации (файл `api.ads`).

Затем в организационных целях мы предоставляем имя базового пакета для всех сгенерированных классов Java, используя флаг -b. Здесь все сгенерированные классы и пакеты будут дочерними элементами пакета Java «test».

Флаги -o и -c определяют каталог, в котором будут храниться, соответственно, Ada и Java-код. Здесь мы помещаем код Ada в каталог `ada/` и java-код в каталог `java/`.

Наконец, основной код Ada (здесь, пакет `My_Package`) и сгенерированный код связующего звена должны быть скомпилированы в совместно используемой библиотеке, которая будет загружена Java. Чтобы упростить процесс, возможно, генерировать файл проекта GNAT для ввода к gprbuild, который обработает компиляцию этой библиотеки, при помощи переключателя –L.

Написание кода Java

Если вы посмотрите на содержимое каталога `java/`, вы найдете кучу классов, в частности:

```
test.API.R
test.API.API_Package
test.Standard.AdaString
```

Это все классы, необходимые нашему приложению. `test.API.R` сопоставляется с объектом Ada `R`, `test.API.API_Package` содержит все объявления подпрограмм, которые поступают из пакета `API` и не могут быть определены как члены других типов, а `test.Standard.AdaString` - автоматически сгенерированный класс, который отображает стандартный тип `String`.

Обратите внимание, что класс `test.API.R` поставляется с аксессуарами и модификаторами для полей `F1` и `F2` и включает конструктор по умолчанию. Также обратите внимание, что `test.Standard.AdaString` поставляется с конструктором на основе `java.lang.String`.

Настало время написать основной класс Java:

```
import test.API.R;
import test.API.API_Package;
import test.Standard.AdaString;

public class My_Main {

public static void main (String [] argv) {
   R v = new R ();
   v.F1 (10);
   v.F2 (15);
  API_Package.Print (new AdaString ("Hello"), v);
}

}
```

Вот и все - это работает так же, как если бы вы написали вызов непосредственно из Java в Ada!

Компиляция и запуск кода

Последним шагом является компиляция и запуск кода. Поскольку мы уже создали файл проекта GNAT для кода Ada, для его компиляции требуется только запуск gprbuild:

```
gprbuild –p –P ada/my_lib.gpr
```

Теперь вам нужно адаптировать `PATH` (в Windows) или `LD_LIBRARY_PATH` (в Linux / Solaris), чтобы перечислить каталог ada / lib, например:

```
LD_LIBRARY_PATH=`pwd`/ada/lib/:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

Сгенерированный Java-код должен быть в вашем `CLASSPATH`, а также в основной подпрограмме, которую вы написали, и в файле `lib/ajis.jar` вашей установки AJIS. Предположим, что вы работаете в Linux, а ваш Java-сервер находится в том же месте, что и файл `api.ads`. Вам понадобиться:

```
CLASSPATH=`pwd`:`pwd`/java:<your AJIS installation>/lib/ajis.jar:$CLASSPATH
```

Компиляция и запуск кода Java теперь просто обычным образом:

```
javac My_Main.java
java My_Main
```

Обратите внимание на то, что библиотека, которую мы скомпилировали, будет автоматически загружена сгенерированным связанным кодом.

Предстоящие Gems покажут более расширенное использование GNAT-AJIS, включая использование обратных вызовов, OOP, исключений и управления памятью.

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

Quentin Ochem имеет опыт разработки программного обеспечения, специализирующийся на разработке программного обеспечения для критически важных приложений. Он имеет более чем 10-летний опыт разработки Ada. Сегодня он работает техническим менеджером AdaCore по проектам, связанным с авионикой, железнодорожной, космической и оборонной отраслями. Он также обучает стандарту DO-178B авионики в университете EPITA в Париже.

Last Updated: 10/13/2017
Posted on: 12/8/2008

Обсуждение…

2 ответов на "Gem # 55: Введение в Ada / Java Interfacing"

1. 9 декабря 2008 года в 21:02

Paul F. Pearson сказал:

Будет ли ada2java поддерживать предложения представления? Например, задание местоположений байтов / бит в записи?

2. 10 декабря 2008 года в 2:12 утра

Quentin Ochem сказал:

Фактически данные остаются на внутренней стороне. Другими словами, здесь инструкция java «new R ()» автоматически создаёт объект типа R на стороне Ada, который будет использоваться для создания собственных вызовов. Если этот тип R имеет предложения представления, они будут применяться, как если бы мы создали экземпляр вручную.