Краткое содержание

Содержание

Предисловие

> Язык JRuby – это просто Ruby, воспользовавшийся преимуществами виртуальной машины Java. Он взял все лучшее от Java и дополнил Ruby целым рядом фантастических возможностей.
>
> *– Чарльз Наттер,*
> *руководитель проекта JRuby,*
> *Twitter, 7 августа 2008*

В этом высказывании Чарльз Наттер кратко выразил оба аспекта, которые в последнее время привлекли внимание к проекту JRuby: во-первых, тот факт, что Java – это отличная платформа, пригодная для самых разных языков, а не только собственно для Java, а, во-вторых, рост интереса к языку программирования Ruby. В этой книге мы рассмотрим множество возможных вариантов использования JRuby. Ели говорить словами Чарльза, то в одних рецептах речь пойдет о том лучшем, что взято от Java, в других – о фантастических возможностях, привнесенных в Ruby, а в третьих – о том и другом вместе.

Предполагаемая аудитория

Если вы хотите получить от JRuby максимум возможного, то должны свободно переходить от Java к Ruby и обратно. Работая над этой книгой, мы ориентировались на читателя, который в той или иной мере владеет обоими языками, хотя, быть может, одним лучше, чем другим. Поэтому обширного введения вы здесь не найдете, разве что в первой главе, где мы описываем, в каких вопросах Ruby и Java похожи, а в каких – различаются.

Вообще говоря, мы ставили себе целью не рассказать о том или ином уже имеющемся в Ruby или Java средстве, а объяснить, как JRuby позволяет воспользоваться имеющимися средствами или расширить их. Так, рецепты в главе, посвященной JRuby on Rails, рассчитаны на читателя, который уже создал хотя бы одно (работающее) приложение Rails.

Организация материала

Глава 1. *Приступая к работе*

Эта глава представляет собой краткое введение в JRuby, за которым следует описание ряда простейших приемов работы с этим языком, в том числе применение системы управления пакетами RubyGems и техника обращения к Java-коду из программы на Ruby. В конце главы приводятся рецепты, касающиеся установки и настройки различных интегрированных сред разработки (IDE) для работы с JRuby.

Глава 2. *JRuby on Rails*
Эта глава посвящена разнообразным сценариям развертывания приложений Ruby on Rails с использованием JRuby.

Глава 3. *Интеграция с Java*
Эта глава начинается несколькими рецептами о вызове Ruby-кода из Java-программы. Затем следуют рецепты, описывающие обращение из Ruby к таким популярным Java-библиотекам, как Java Native Access (JNA) и Jakarta Commons Logging.

Глава 4. *Java в системах масштаба предприятия*
Все рецепты, собранные в этой главе, относятся к использованию JRuby в таких предназначенных для создания крупномасштабных систем каркасах, как JMS, JNDI, EJB, Spring и Hibernate.

Глава 5. *Пользовательский интерфейс и графические приложения*
В этой главе описываются некоторые основанные на JRuby каркасы для создания графических пользовательских интерфейсов. Сюда же включены рецепты, касающиеся операций над графическими изображениями, аплетов и интеграции с рабочим столом.

Глава 6. *Инструменты сборки*
В этой главе речь пойдет об использовании JRuby для усовершенствования процедуры сборки Java-проектов. Оба наиболее популярных в мире Java инструмента сборки – Ant и Maven – допускают различные способы подключения JRuby. Здесь же вы найдете рецепты, посвященные программам Raven и Buildr, которые ориентированы исключительно на JRuby.

Глава 7. *Тестирование*
Эта глава посвящена главным образом пакету JtestR, который включает в себя JRuby и ряд популярных инструментов тестирования для языка Ruby. Содержащиеся здесь рецепты научат вас писать на Ruby тесты для Java-программ.

Глава 8. *Сообщество пользователей JRuby*
Это последняя глава. В нее включены сведения о том, как можно принять участие в жизни сообщества пользователей JRuby.

Типографские соглашения

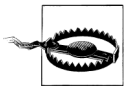
В книге применяются следующие соглашения:

Курсив
Таким начертанием выделяются URL, имена каталогов и файлов, параметры. Иногда оно применяется для выделения важной информации.

`Моноширинный шрифт`
Применяется для листингов программ, а также внутри текста для обозначения элементов программы, как то: имена пространств имен, классов и методов.

Моноширинный курсив
Текст, вместо которого надо подставить значения, вводимые пользователем.

Этот значок обозначает совет, рекомендацию или замечание общего характера.



Этот значок обозначает предупреждение или предостережение.

О примерах кода

Эта книга призвана помогать вам в работе. Поэтому вы можете использовать приведенный в ней код в собственных программах и в документации. Спрашивать у нас разрешение необязательно, если только вы не собираетесь воспроизводить значительную часть кода. Например, не требуется разрешение, чтобы включить в свою программу несколько фрагментов кода из книги. Однако для продажи или распространения примеров на компакт-диске нужно получить разрешение. Можно без ограничений цитировать книгу и примеры в ответах на вопросы. Но чтобы включить значительные объемы кода в документацию по собственному продукту, нужно получить разрешение.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются название книги, имя автора, издательство и ISBN, например: «*JRuby Cookbook,* by Justin Edelson and Henry Liu. Copyright 2009 Justin Edelson and Henry Liu, 978-0-596-51980-3».

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу *permissions@oreilly.com*.

Доступность на Safari

Если на обложке вашей любимой книги присутствует значок Safari® Enabled, это означает, что книга доступна он-лайн в сетевой библиотеке Safari издательства O'Reilly. У Safari есть преимущество перед обычными электронными книгами. Это виртуальная библиотека, которая позволяет легко находить тысячи технических книг, копировать примеры программ, загружать отдельные главы и быстро получать точную и актуальную информацию. Бесплатный доступ по адресу *http://safari.oreilly.com*.

Вопросы и замечания

Мы тщательно проверили информацию, приведенную в этой книге, и протестировали код. Однако полностью исключить ошибки и опечатки невозможно. Если вы

обнаружите какую-нибудь ошибку или захотите высказать пожелание для будущих изданий, пожалуйста, обращайтесь по адресу:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (в США или Канаде)
707-829-0515 (международный или местный)
707-829-0104 (факс)

Замечания и вопросы технического характера следует отправлять по адресу:

bookquestions@oreilly.com

Для этой книги есть веб-страница, на которой выкладываются списки замеченных ошибок, примеры и планы для будущих изданий. Адрес страницы:

http://www.oreilly.com/catalog/9780596519803

Дополнительную информацию о наших книгах и прочих предприятиях можно найти на сайте:

http://www.oreilly.com

Благодарности

Мы благодарны коллективу издательства O'Reilly, особенно нашему редактору Майку Лукидесу (Mike Loukides) и литературному редактору Коллин Горман (Colleen Gorman). Спасибо также Стивену Шинглеру (Steven Shingler) за его вклад в главу 4. И всем рецензентам этой книги: Хуану Пабло Таркино (Juan Pablo Tarquino), Джону Пэрселлу (John Purcell) и Дэвиду Кунтцу (David Koontz).

Эта книга никогда не появилась бы на свет без неустанных трудов всей команды, работающей над проектом JRuby, а именно: Чарльза Наттера (Charles Nutter), Томаса Энебо (Nick Sieger) и Ола Бини (Ola Bini). Благодарим также корпорацию Sun и компанию ThoughtWorks за постоянную поддержку JRuby. Проект JRuby размещен на сайте The Codehaus; мы благодарны его создателю Бобу Маквиртеру (Bob McWhirter).

Мы также признательны Нику Рокуэллу (Nick Rockwell), который не перестает подбадривать и воодушевлять нас.

Justin Edelson

Эта книга не состоялась бы, если бы не любовь и поддержка моей чудесной жены Элизабет. Отдельное спасибо моим сыновьям: Оуэну, который сам напечатал свое имя, и Бенджамину, который пока еще этого не умеет.

Благодарю своих коллег в компании MTV Networks: Майкла Бенуа (Michael Benoit), Кита Гриффина (Keith Griffin), Рамеша Нуталапати (Ramesh Nuthalapati), Илью Резникова (Ilya Reznikov), Криса Синдела (Chris Sindel), Джеффа Йемина (Jeff Yemin) и Чжун Чжоу (Jun Zhou) за достойную работу.

Спасибо также Уоррену Хабибу (Warren Habib) за поддержку.

Генри Лю

Спасибо моему другу Йону Баэру (Jon Baer), который впервые привел меня на мероприятие, посвященное языку Ruby, и на протяжении многих лет составляет мне компанию в работе. Я благодарен Фрэнсису Хуангу (Francis Hwang), Мэтту Пеллетье (Matt Pelletier), Себастьяну Дельмонту (Sebastian Delmont), Троттеру Кэшину (Trotter Cashion) и всем остальным участникам группы пользователей Ruby в Нью-Йорке. Они учили меня Ruby и Rails, отвечали на мои наивные вопросы. Именно их увлеченность этой технологией и побудила меня копать глубже. Благодарю всех своих коллег по компании MTV Networks и в особенности Марка Эйка (Mark Ache), Люка Мэрфи (Luke Murphy) и Стива Азуэту (Steve Azueta) за непрестанную поддержку. А больше всех благодарю свою семью и подругу Наоми; без нее ничего с этой книгой не получилось бы.

Глава 1

Приступая к работе

1.0. Введение

JRuby – это реализация языка программирования Ruby для виртуальной машины Java (JVM), распространяемая с открытым исходным кодом. Она позволяет запускать Ruby-приложения на виртуальной машине и обращаться к библиотекам, написанным как на Java, так и на Ruby. Работа над проектом JRuby началась еще в 2001 году, но за последние несколько лет интерес к нему значительно возрос, что отражает общий рост интереса к Ruby, обусловленный успехом каркаса Ruby on Rails. Корпорация Sun немало содействовала популяризации JRuby в частности тем, что приняла в свой штат основных разработчиков и включила поддержку JRuby в среду разработки NetBeans. В настоящее время официальный сайт проекта JRuby расположен по адресу *http://www.jruby.org*.

Ruby

Ruby – это динамический объектно-ориентированный язык, автором которого является Юкихиро Мацумото (Yukihiro Matsumoto), известный под псевдонимом Мац (Matz). Язык был создан в середине 1990-х годов. При нумерации версий Ruby применяется то же соглашение, что для ядра Linux: четными номерами обозначаются стабильные версии, а нечетными – находящиеся в разработке. Поэтому существуют две *текущие* версии Ruby: 1.8.6, выпущенная в марте 2007 года (текущая стабильная версия), и 1.9.0, выпущенная в декабре 2007 года (текущая разрабатываемая версия). Стандартный интерпретатор Ruby* написан на языке C. Есть также несколько альтернативных реализаций, в том числе JRuby, IronRuby (для Microsoft .NET Framework) и Rubinius. Для Ruby не существует формальной спецификации языка, однако работа в этом направлении ведется на вики-сайте *http://spec.ruby-doc.org*.

Поскольку Ruby является объектно-ориентированным языком, многие базовые идеи знакомы разработчикам на Java, хотя синтаксис, конечно, отличается. Самая существенная особенность Ruby – поддержка *блоков*. В Ruby блоком называется группа предложений, которая передается вызываемому методу. Метод-получатель может вызывать блок несколько раз, передавая ему параметры. Обсуждается включение похожей конструкции, *замыкания*, в версию Java 7. На данный момент представлено несколько конкурирующих предложений; какое из них будет выбрано и будет ли выбрано вообще, пока неясно. В листинге 1.1 на примере простого класса Ruby демонстрируются два способа определения блока. Первый вариант – с фигурными скобками – обычно применяется для создания блока, содержащего одно предложение; второй – с ключевыми словами `do` и `end` – для блоков из нескольких предложений.

Пример 1.1. Введение в блоки Ruby

```
class HelloWorldSayer
    def hello_world
        yield "Hello"
```

* Обычно называется Matz's Ruby Interpeter (MRI).

```
        yield "World"
        yield "from Ruby"
    end
end

sayer = HelloWorldSayer.new
sayer.hello_world { |message| puts message.swapcase }

# или

sayer.hello_world do |it|
  puts it.swapcase
end
```

Функция Ruby `yield` передает управление методу-аргументу блока.

Мы вовсе не хотим сказать, что блоки – единственное заметное различие между Ruby и Java, но оно, безусловно, является одним из самых существенных, поскольку блоки встречаются в программах на Ruby повсеместно. Так, в примере 1.2 приведена Java-программа для распечатки списка целых чисел от 1 до 10. Соответствующий код на Ruby показан в примере 1.3.

Пример 1.2. Цикл в Java

```
for (int i = 1; i <= 10; i++) {
  System.out.println(i);
}
```

Пример 1.3. Цикл в Ruby

```
1.upto(10) { |x| puts x }
```

Вокруг языка Ruby образовалось активное сообщество разработчиков – как в Сети, так и в виде местных групп. На сайте *http://www.ruby-lang.org* имеется дополнительная информация об этих группах пользователей. Языку Ruby посвящено немало книг, из которых наиболее известны знаменитая «Киркомотыга» (по картинке на обложке) – *Programming Ruby*: The Pragmatic Programmers's Guide (Pragmatic Bookshelf) by Dave Thomas, Chad Fowler, and Andy Hunt – и *The Ruby Programming Language* by David Flanagan and Yukihiro Matsumoto (O'Reilly).

JRuby

Первая версия JRuby была написана Яном Арне Петерсеном (Jan Arne Petersen) в 2001 году и явилась прямым переносом интерпретатора Ruby 1.6, реализованного на C. В течение нескольких последующих лет проект оставался интересным, но имел серьезные ограничения в плане производительности. Вслед за выпуском

версии Ruby 1.8 в 2003 году, а затем и каркаса Ruby on Rails для веб-приложений в 2004 разработке JRuby уделялось очень много внимания, особенно в части совместимости и производительности. В сентябре 2006 года корпорация Sun Microsystems по сути дела взяла на себя финансирование проекта, приняв в штат двух ведущих разработчиков, Чарльза Наттера и Томаса Энебо. Позже взяли на работу также третьего разработчика, Ника Сигера*.

Для корпорации Sun язык JRuby – это возможность распространить виртуальную машину Java на новые сферы. Первоначально JVM была очень тесно связана с языком Java, но появление таких проектов, как JRuby, Jython (реализация языка Python на платформе Java), Groovy (язык сценариев, берущий истоки в Ruby) и Scala (одновременно функциональный и объектно-ориентированный язык) доказало, что JVM может стать платформой для самых разных языков. Эта тенденция увенчалась созданием спецификации Java Specification Request (JSR) 223, описывающей технологию сценариев на платформе Java. В документе JSR 223 определен стандартный API (интерфейс прикладного программирования), позволяющий интегрировать сценарные языки с JVM. На сайте *https://scripting.dev.java.net* имеются реализации JSR 223 API для 25 языков программирования. Мы еще вернемся к обсуждению этого API в главе 3.

Для пользователей JRuby открывает возможность воспользоваться всей мощью динамического языка, каковым является Ruby, не отказываясь от существующих библиотек и серверов приложений, написанных на Java. Этот аспект мы будем рассматривать в двух первых главах.

С выходом версии JRuby 1.1 в апреле 2008 года проблема производительности JRuby была решена за счет нового написанного на C интерпретатора Ruby; во многих случаях язык стал работать быстрее. С точки зрения совместимости разработчики JRuby стремятся дублировать поведение стандартного интерпретатора Ruby, даже если для этого приходится поступаться согласованностью с принципами Java. Включена большая часть базовых классов Ruby, а также многие классы из стандартной библиотеки Ruby, система управления пакетами RubyGems, система документирования RDoc и система сборки Rake. Несмотря на прилагаемые усилия, в некоторых областях поведение JRuby все же отличается от стандартного интерпретатора Ruby. Самый наглядный пример – работа с потоками в JRuby. В этом отношении JRuby на самом деле ушел вперед от стандартного интерпретатора. Потоковая модель, которую JRuby поддерживает уже сейчас, предположительно будет включена только в версию Ruby 2.0.

В этой главе мы рассмотрим процедуру установки JRuby, приведем кое-какую информацию об интеграции Java и Ruby и обсудим разнообразные настройки IDE, относящиеся к интеграции.

---

* Четвертый из основных разработчиков, Ола Бини, трудится во влиятельной консалтинговой ИТ-компании ThoughtWorks.

1.1. Установка JRuby

Задача

Требуется установить JRuby.

Решение

Скачайте последнюю двоичную версию с сайта JRuby *http://www.jruby.org* и распакуйте архив. Добавьте каталог *bin* в переменную окружения `PATH`.

Обсуждение

Windows

На сайте JRuby имеются двоичные дистрибутивы в форматах ZIP и TGZ. Начиная с Windows XP, средства для работы с ZIP-архивами уже встроены в операционные системы Windows. Существуют также открытые и коммерческие программы, поддерживающие формат TGZ, например: WinZip (*http://www.winzip.com*), 7-Zip (*http://www.7-zip.org*) и IZArc (*http://www.izarc.com*).

JRuby может размещаться в любом месте. Лично я предпочитаю устанавливать библиотеки и исполняемые файлы Java в подкаталоги каталога `C:\java`. На рис. 1.1 показан результат распаковки двоичного дистрибутива последней версии (во время работы над книгой текущей была версия 1.1).

Сразу после распаковки можно начинать работать с JRuby. Полюбоваться на JRuby в действии проще всего, запустив программу `jirb`, вариант Interactive Ruby (`irb`) для JRuby. Как и `irb`, `jirb` позволяет вводить предложения языка Ruby и

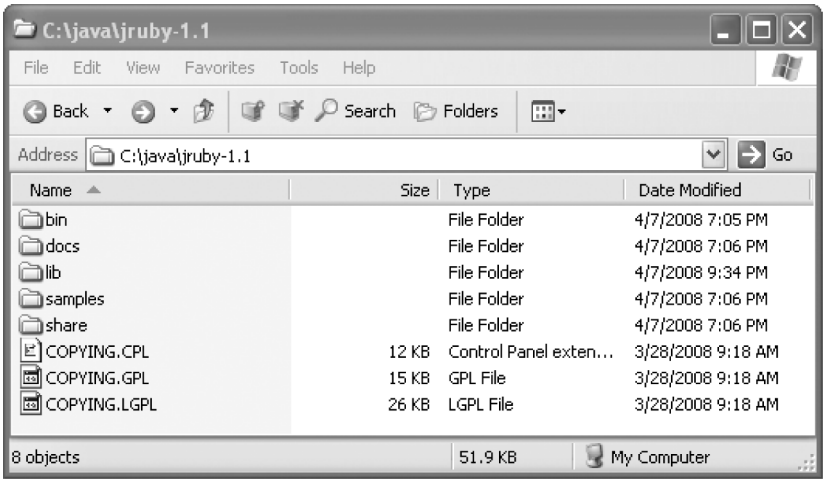


Рис. 1.1. Распакованный двоичный дистрибутив JRuby

сразу же видеть результат их выполнения. В каталоге *bin* имеются командная и графическая версии `jirb`. Для запуска командной версии (рис. 1.2) следует выполнить файл *bin\jirb.bat*, а для запуска графической (рис. 1.3) – файл *bin\jirb_swing.bat*. На обоих рисунках показаны результаты выполнения тривиального Ruby-кода. Вы видите как строку, которую печатает метод `puts` (`Hello World`), так и возвращаемый этим методом результат (`nil`).

Рис. 1.2. Командная версия jirb

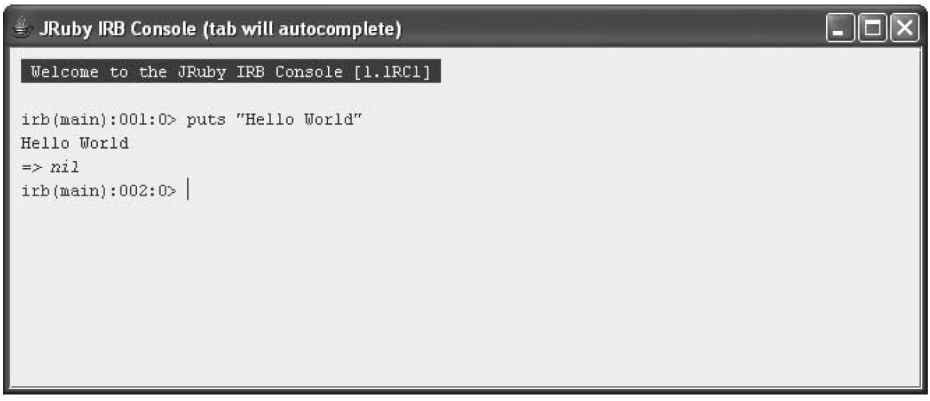
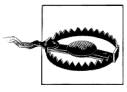


Рис. 1.3. Графическая версия jirb



Если при запуске *jirb.bat* или *jirb_swing.bat* из Windows Explorer вы видите, как появляется и тут же пропадает черное окошко, то, вероятно, не установлена или неправильно установлена переменная среды `JAVA_HOME`. Для установки переменных среды в Windows откройте минприложение **Система** на панели управления и перейдите на вкладку **Дополнительно**. Переменная `JAVA_HOME` должна указывать на каталог, в который установлена Java.

Можно также протестировать JRuby из командной строки, задав параметр `-e` (evaluate):

```
C:\java\jruby-1.1\bin\jruby -e "puts 'Hello World'"
```

Чтобы не вводить каждый раз полный путь к каталогу *bin* JRuby, добавьте его в переменную среды `PATH`. Для этого откройте миниприложение **Система** на панели управления и перейдите на вкладку **Дополнительно**. Нажмите кнопку **Переменные среды**. Откроется диалоговое окно «Переменные среды», показанное на рис. 1.4. С помощью кнопок **Создать** и **Изменить** в секции **Системные переменные** добавьте переменную среды `JRUBY_HOME` и допишите в начало переменной `PATH` путь `%JRUBY_HOME%\bin`. Можно вместо этого просто дописать в начало `PATH` полный путь к каталогу *bin*, но при наличии отдельной переменной среды будет проще переходить на новую версию.

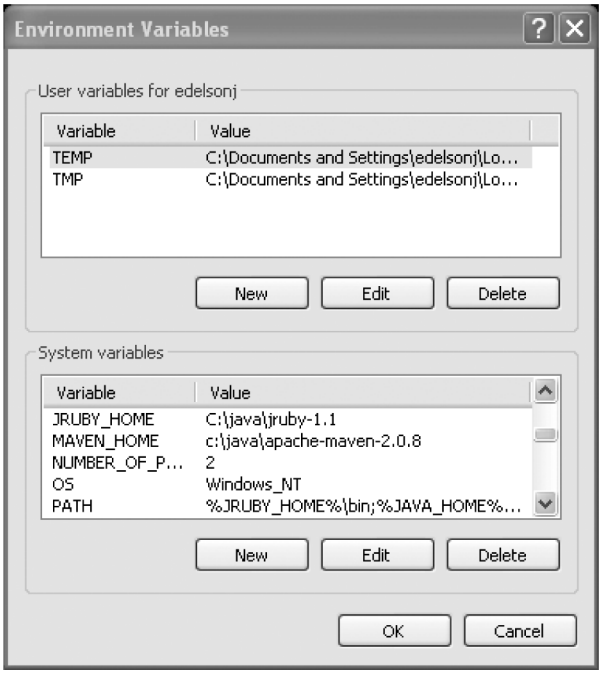


Рис. 1.4. Диалоговое окно «Переменные среды»

Linux и Mac OS X

На сайте JRuby имеются двоичные дистрибутивы в форматах ZIP и TGZ. Хотя в большинстве дистрибутивов Linux и в OS X есть утилиты для работы с архивами обоих типов, предпочтительно все же использовать TGZ-файл, так как в нем сохраняется информация о правах доступа (в ZIP-архиве она отсутствует).

На сайте JPackage Project по адресу *http://www.jpackage.org* имеется дистрибутив в формате RPM. Во время работы над этой книгой последней версии JRuby там еще не было, но, когда вы будете ее читать, ситуация может измениться.

Если у вас есть привилегии суперпользователя root в той системе, где устанавливается JRuby, то установку следует производить в общепринятые каталоги. В зависимости от конкретной ОС это может быть каталог */usr/local/jruby*, */usr/share/jruby*, */opt/JRuby* или еще какой-нибудь. В системе Mac OS X пользователям рекомендуется производить установку пакетов в каталоги */opt/local/jruby* или */usr/local/jruby*. Если привилегий root у вас нет, то устанавливать, скорее всего, придется в подкаталог собственного начального каталога, например *~/jruby*. По умолчанию версии JRuby распаковываются в каталог, имя которого включает номер версии, поэтому мы создадим символическую ссылку с *~/jruby* на *~/jruby-1.1*. Это упростит переход на следующую версию в будущем:

```
$ cd ~
$ tar -xzf jruby-bin-1.1.tar.gz
$ ln -s jruby-1.1 jruby
```

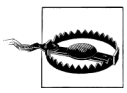
Создайте переменную среды `JRUBY_HOME`, указывающую на инсталляционный каталог, и добавьте путь к каталогу *bin* JRuby в переменную среды `PATH`; вставьте в файл *~/.profile* строки, показанные в примере 1.4.

Пример 1.4. Добавление пути к JRuby в переменную среды PATH с помощью файла .profile

```
export JRUBY_HOME=~/jruby
export PATH=$JRUBY_HOME/bin:$PATH
```

Добавив путь в каталогу *bin* в переменную среды PATH, вы можете проверить правильность установки, попытавшись выполнить такой Ruby-сценарий:

```
$ jruby -e "puts 'Hello World'"
Hello World
```



Чтобы воспользоваться какой-либо командной утилитой, входящей в дистрибутив JRuby, в частности `jirb`, необходимо предварительно добавить путь к каталогу *bin* в переменную среды `PATH`.

См. также

- Рецепт 8.1 «Сборка JRuby из исходных кодов»
- Рецепт 1.3 «Одновременное использование Ruby и JRuby»

1.2. Управление пакетами с помощью RubyGems

Задача

Требуется установить Ruby on Rails или другие Ruby-пакеты для использования совместно с JRuby.

Решение

Воспользуйтесь встроенной в JRuby поддержкой системы RubyGems. Начать работу с ней можно сразу же после установки JRuby, достаточно запустить сценарий *gem*, находящийся в каталоге *bin* JRuby. Для установки пакета выполните следующую команду:

```
$ gem install packagename
```

Например, чтобы установить каркас Ruby on Rails, выполните команду:

```
$ gem install rails
```

Обсуждение

RubyGems – это стандартная система управления пакетами и подготовки дистрибутивов, ориентированная на пакеты Ruby. В репозитории RubyGems по адресу *http://gems.rubyforge.org* есть тысячи таких пакетов (они называются *gem-пакетами*), большинство совместимы с любой реализацией Ruby.

К числу наиболее употребительных команд RubyGems относятся `install`, `query`, `update`, `uninstall` и `rdoc`. Полный список можно получить с помощью команды `help`*:

```
$ gem help commands
Список команд GEM:
build              Создать gem-пакет по gem-спецификации
cert               Управление параметрами сертификатов и электронной
                   подписью RubyGems
check              Проверить установленные gem-пакеты
cleanup            Удалить старые версии gem-пакетов из локального
                   репозитория
contents           Вывести содержимое установленных gem-пакетов
dependency         Показать зависимости установленного gem-пакета
```

* Для удобства читателя справка переведена на русский язык. *Прим. перев.*