DZone Refcardz

Essential **Ruby**

By Melchior Brislinger and Peter Cooper

ABOUT RUBY

Ruby is an easy-to-learn, dynamic, object-oriented programming language with dynamic typing and automatic memory management. While object-oriented at heart, it provides facilities for procedural and functional programming as well as extensive support for introspection and meta-programming. Ruby's core API, extensive standard library, and thousands of high-quality external libraries make it suitable for many different programming tasks in multiple disciplines (key examples being network programming, Web applications, shell scripts, data processing, and text manipulation).

Ruby is already installed on Mac OS X and many Linux distributions. For Windows the easiest way to install everything necessary is the Ruby Installer (http://rubyinstaller.rubyforge.org).

This refcard provides a quick reference to language elements and many important API functions for quick lookup.

RUBY LANGUAGE OVERVIEW

Ruby is considered to be a "pure" object-oriented language because almost every concept within Ruby is object-oriented in some sense. Yukihiro "Matz" Matsumoto, Ruby's creator, wanted to develop a language that operated on the "principle of least surprise" meaning that code should behave in a non-confusing manner and be reasonably self-explanatory (beyond the basic syntax). Matz also wanted Ruby to be a pleasurable language with which to program, and not make unnecessary demands upon the programmer.

Ruby is considered a "reflective" language because it's possible for a Ruby program to analyze itself (in terms of its make-up), make adjustments to the way it works, and even overwrite its own code with other code. It's also considered to be "dynamically typed" because you don't need to specify what type of objects can be associated with certain variables. Objects are considered prime in Ruby and whether you're passing around a string, a number, a regular expression, or even a class, you're just dealing with an object from Ruby's point of view.

Ruby will seem reasonably familiar to Python and Perl programmers (and to a lesser extent C# and JavaScript developers) as Ruby was heavily inspired by Perl in certain areas (as was Python). Ruby is less similar to languages like C, C++ or Java because these languages are compiled (not interpreted), statically typed, and focused on performance rather than flexibility and conciseness.

SIMPLE RUBY EXAMPLES

Despite being an object-oriented language, it is not necessary to use explicitly object-oriented syntax within a basic Ruby program. While everything works on objects (and methods called upon those objects) behind the scenes, you can write a program as simply as this:

```ruby
def fib(i)
  if i.zero?
    0
  elsif i == 1
    1
  else
    fib(i - 2) + fib(i - 1)
  end
end

puts fib(10)
```

This script prints to screen the 10th number in the Fibonacci sequence. It defines a method called `fib` that returns the relevant result from a simple `if/elsif/else` expression. Note the use of standard equality syntax (`==`), addition (`+`), subtraction (`-`), and method calling (`fib(10)`), but also note the possibility of using methods in somewhat idiomatic ways (`i.zero?` rather than `i == 0`—though the latter would also work). The use of `i.zero?` demonstrates calling a method upon an object (where `i` is the object, and `zero?` is the method).

Hot Tip

The main Ruby interpreter is usually invoked by running "ruby" from the command line. If it is given a filename as an argument that file will be run (e.g. `ruby myscript.rb`). The interpreter has several other options that are listed in the "Ruby Interpreter Arguments" table in this card's reference section.

Simple Ruby Examples, continued

Developing a program with "true" object-oriented syntax is not significantly different. For example:

```ruby
class Person
  attr_accessor :name, :age

  def full_info
    return "#{@name} is #{@age} years old"
  end
end

fred = Person.new
fred.name = "Fred"
fred.age = 45
puts fred.full_info
```

In this example, a class (`Person`) is defined, and attributes (`name` and `age`) and a method (`full_info`) are defined upon that class. Below the class definition, we then create an instance of the `Person` class and assign it to a variable, `fred`, before assigning values to its attributes, and then calling that instance's `full_info` method (which, in turn, uses instance variables—prefixed with @—to create its output).

> **Hot Tip**
>
> "This is a test" is a string with no special qualities (and, remember, it's also an object in Ruby) but it's possible to interpolate data into it (from variables, etc.) with a special syntax:
>
> `"2 plus 2 is #{2 + 2}"`
>
> The #{} construction serves to interpolate the result of the expression within the curly braces—in this case 2 + 2 is calculated to equal 4 and so the string ends up as *2 plus 2 is 4*

Earlier we called Ruby a "reflective" language because it offers functionality to programs to change, extend, and otherwise inspect themselves. We can look at a key Ruby idiom and reflective feature—class reopening—by changing the Fibonacci example from earlier to the following:

```ruby
class Integer
  def fib
    if self.zero?
      0
    elsif self == 1
      1
    else
      (self - 2).fib + (self - 1).fib
    end
  end
end

puts 10.fib
```

Note this time that in order to get the Fibonacci number, we're no longer calling a global `fib` method, but a method that works directly upon the number 10 itself (remember, everything is an object—even the number 10!). The way this is achieved is by "reopening" a standard Ruby class—`Integer`—and defining a new method called `fib` within it. This then makes the `fib` method available to all objects of class `Integer` in Ruby! Note that the content of the integer object itself (the number we need to use) is obtained with the `self` keyword. `self`, in this case, returns a representation of the current object in its native form. In this sense, Ruby is very similar to Python.

IRB

IRB (short for "Interactive Ruby") is an interactive prompt or "Read-Eval-Print-Loop" (REPL) that uses the Ruby interpreter. Anything you type is evaluated by Ruby and the response printed to screen. IRB can be invoked by running "irb" from the command. A demonstrative session shows the usage:

```
irb(main):001:0> 3 + 5
=> 8

irb(main):002:0> "hello there " * 3
=> "hello there hello there hello there "

irb(main):001:0> "A String".class
=> String

irb(main):002:0> "A String".methods.sort
=> ["%", "*", "+", "<", "<<", "<=", "<=>", "==",
"===", "=~", ">", ">=", "[]", "[]=", "__id__", "__
send__", "all?", …

irb(main):003:0> "A String".class.methods.sort
=> ["<", "<=", "<=>", "==", "===", "=~", ">", ">=",
"__id__", "__send__", "allocate", "ancestors",
"autoload", ...
```

IRB is most commonly used when learning the Ruby programming language, and also as a handy "sand box" to try out new programming tricks and techniques quickly. IRB can be used to interactively explore classes, test pieces of code and is also used as a console to inspect and manipulate running programs, for example, in Web applications.

> **Hot Tip**
>
> Want to try Ruby without installing anything? Or want to get a walkthrough tutorial? Go to **http://tryruby.hobix.com**. It's a Web-based version of IRB and Ruby, and features a tutorial to bring you up to speed.

RUBYGEMS

RubyGems is the official Ruby package manager (though, notably, it is not included with default Ruby 1.8 releases by default—although it is present within Ruby 1.9 and on the OS X version of Ruby 1.8). It allows developers and users to easily search, install and update Ruby libraries and their dependencies and works in a similar fashion to other package management tools (such as yum and apt-get).

Gems are installed by running "gem install" and the name of the gem (e.g. gem install rails). Running "gem update" updates all installed gems to their latest official versions.

A selection of popular Ruby gems/libraries:

gem/library	Description	URL
Rails	The famous Web application framework	http://www.rubyonrails.com
Rake	A Ruby based build system (like a Ruby equivalent of make)	http://rake.rubyforge.org
Capistrano	A tool for automatic remote deployment tasks	http://capify.org
Mongrel	A Ruby Web server and HTTP daemon library	http://mongrel.rubyforge.org
rspec	A "Behavior Driven Development" (BDD) framework	http://rspec.info
camping	A tiny web framework	http://code.whytheluckystiff.net/camping

Information about RubyGems can be found at:
http://www.rubygems.org

RUBY LANGUAGE REFERENCE TABLES

The following reference tables provide a quick look at many elements of Ruby's syntax. These can be used as a comparison to other languages in order to see how the syntax differs. Ruby's syntax is often rather different to that of, say, Java or C#.

Types

123	Integer (Fixnum or Bignum)
12345 1.23e-4	Float
0xFF00 0b01100 0244	Integer as hexadecimal, binary, or octal
1..5 'a'..'z'	Range (inclusive
1...5 'a'...'z'	Range (non-inclusive – e.g. 1...5 represents 1 through 4)
?c	Character
'string'	String
"string\n"	Double-quoted String with escape character
"string #{...}"	Double-quoted String with inline expressions
<<DOC *string* *DOC*	Heredoc String
:symbol	Symbol
/regexp/opts	Regexp (regular expression)
[123, 'string', object, :symbol]	Array
{1 => 2, :symbol =>'string' }	Hash (associative array)

Literals

%q %Q(string)	Single/double-quoted String
%w %W(string string string)	Array of Strings (no quotes for the Strings)
%r(regexp)	Regexp (regular expression)

Variables

local	Locally scoped variable
@instance	Instance scoped variable
@@class	Class scoped variable
$global	Globally scoped variable
Constant	Constant

Expressions

if *condition*
  ...
end
	while *condition*
	  ...
	end

if *condition*
  ...
elsif *condition*
  ...
else
  ...
end
	until *condition*
	  ...
	end

	do ... while *condition*

unless *condition*
  ...
else
  ...
end
	do
	  ...
	until *condition*

... if *condition*	for *object* in *enumerable* ... end
... unless *condition*	
condition ? ... : ... (a ternary operator)	break next redo retry

case ...
when *condition*
  ...
else
  ...
end
	yield *arguments*

Modules & Classes

module *Name* ... end	Defines a module
class *Name* < *Super* ... end	Defines a class with a superclass
class << *SomeClass* ... end	Defines/accesses the singleton class of *SomeClass*— suited for defining class methods rather than instance methods
include *Module*	Includes module in class
def *name(arguments)* ... end	Defines instance method

Modules & Classes, continued

|	

def *Class.name(arguments)*
  ...
end
 	
def self.*name(arguments)*
  ...
end
	Defines class method

public protected private	Methods below are public/protected/private
public *symbol* protected *symbol* private *symbol*	Methods with names supplied as symbols are public/protected/private
attr *symbols* attr_accessor *symbols* attr_reader *symbols* attr_writer *symbols*	Creates accessor methods for all variables given
alias :*new_method_name* :*method_name*	Creates alias for method with name
super(*arguments*)	Calls same method of superclass

Constants

__FILE__	Filename of current source code file
__LINE__	Current line
__END__	End of Ruby code (ignore everything below)
DATA	Anything below __END__ as an IO/File object
ENV[]	Environment Variables
ARGV[] ARGF[]	Command Line Arguments

Exceptions

begin
  ...
rescue *exception* => *variable*
  ...
else
  ...
ensure
  ...
end
	Try a block of code and catch possible exceptions

Figure 1 shows the Exception hierarchy.
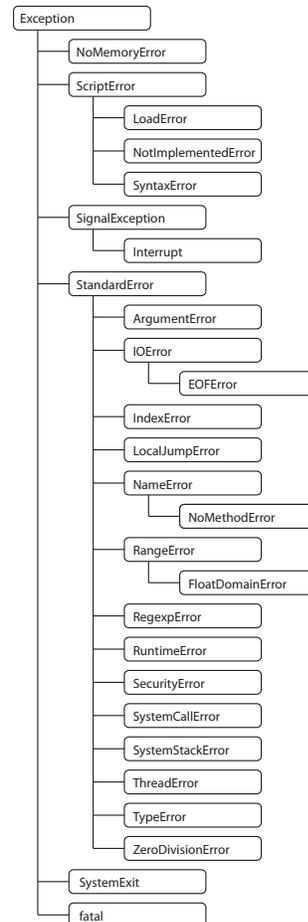
Ruby Language Reference Tables, continued

Ruby Tools

|	
ruby	The Ruby interpreter
irb	An interactive Ruby prompt
ri *symbol*	Shows documentation for the specified symbol
rdoc	Generates HTML documentation form Ruby source files
gem	RubyGems, the Ruby package manager—not always available by default

Ruby Interpreter Arguments

-c	Check code
-d	Debug
-e "…"	Execute a single line expression
-h	Help
-n	gets loop
-r*Library*	`require` the specified library
-v	Verbose mode
-w	Display code warnings
-y	Enable compiler debug mode
-rubygems	Loads RubyGem support

Regular Expressions

.	Any character (excluding newlines)
[…]	Any single character in set
[^…]	Any single character not in set
*	Zero or more
+	One or more (to as many as possible)
+?	One or more (to as few as possible)
?	Zero or one
| (pipe symbol)	Alternatives (e.g. a|b|c will match a, b, or c)
(…)	Group
^	Beginning of line or string
$	End of line or string
{*n, m*}	n to m (as a quantity)
(?>…)	Atomic group
(?=…)	Lookahead
(?!…)	Negative lookahead
\N	Back reference N (where N is a digit)
\A	Beginning of a string
\b	Word boundary
\B	Non-word boundary
\d	Digit
\D	Non-digit
\s	Whitespace
\S	Non-whitespace
\w	Word-character (alphanumeric)
\W	Non-word-character
\z	End of a string
\Z	End of string, before newline
/…/imx	Case insensitive, multiline, ignore whitespace

Ruby Core API

Figure 2 shows important Core API classes and their inheritance tree.
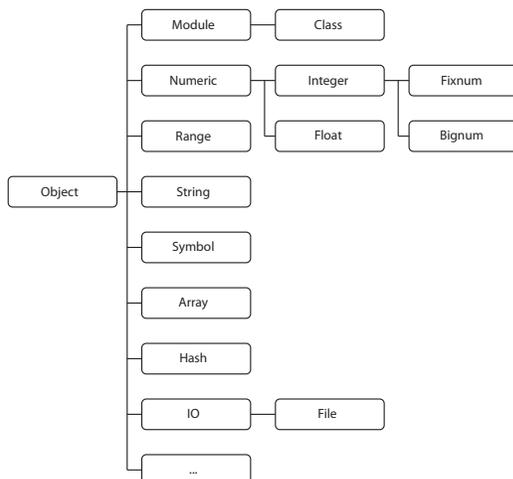
Ruby Core API, continued

The following is a selection of important Ruby Core API objects and methods. Instance methods are written .method and called object.method while class methods are written #method and called Class.method.

Object

.class		Returns the object's class
.inspect		Returns a string containing information about the object
.instance_eval .instance_eval { … }	String *code* Block	Evaluates a string or block in the context of the object
.is_a? .kind_of?	Class *class* Class *class*	Returns *true* if the object's class equals the argument
.methods		Returns an array with the object's methods
.nil?		Returns *true* if the object equals *nil*
.respond_to?	Symbol *methodname*	Returns *true* if the object responds to the method
.send	Symbol *methodname*, *[arguments]*	Sends the message to the object along with optional arguments
.to_s		Returns a string of the object

Enumerable

.all? { |*object*| … }		Sends all elements to the block and returns *true* if every block returned *true*
.any? { |*object*| … }		Sends all elements to the block and returns *true* if *any* block returned *true*
.map { |*object*| … }		Sends all elements to the block and returns a new *Array* with each result
.find { |*object*| … } .detect { |*object*| … }		Sends all elements to the block and returns the first element for which the blocks result is not *false*
.find_all { |*object*| … } .select { |*object*| … }		Sends all elements to the block and returns all elements for which the block is not *false*
.grep	Object *pattern*	Returns a new *Array* with all elements for which *pattern === element*
.include?	Object *object*	Returns *true* if the collection includes *object*
.sort [{|*object, object*| … }]		Returns the *Array*, sorted by each elements <=> or by the block

Array (Enumerable)

[] [] []	Fixnum *index* Fixnum *start*, Fixnum *length* Range *range*	Returns the object at the specified *index* or all objects in the specified *range*
.compact		Returns the *Array* without element that equal *nil*
.delete	Object *object*	Deletes *object* from the *Array*
.delete_at	Fixnum *index*	Deletes the object at *index* from the *Array*
.delete_if { |*object*| … }		Deletes elements for which the block returns *true*
.each { |*object*| … }		Sends each element to the block
.flatten		Flattens the *Array*
.index	Object *object*	Returns the index of the first occurrence of *object*
.insert	Fixnum *index*, Object *object*	Inserts *object* at the position specified by *index*
.join	String *separator*	Returns a *String* with all elements separated by *separator*
.length		Returns the number of elements
.pop		Returns the last element and removes it
.push	Object *object...*	Pushes *object* to the end of the *Array*
.reverse		Reverses the order of elements
.rindex	Object *object...*	Returns the index of the last occurrence of *object*
.shift		Returns the first element and removes it
.uniq		Returns a new *Array* without duplicates
.unshift	Object *object...*	Pushes *object* to the front of the *Array*

Ruby Core API, continued

Hash (Enumerable)

[]	Object *key*	Returns the value for *key*
[] = *value*	Object *key*	Sets the value for *key*
.delete	Object *key*	Deletes *key* and *value* from the *Array*
.delete_if { |*key, value*| … }		Deletes *key* and *value* for which block returns *true*
.each { |*key, value*| … }		Sends each *key* and *value* to the block
.each_key { |*key*| … }		Sends each *key* to the block
.each_value { |*value*| … }		Sends each *value* to the block
.include? .key?	Object *object...* Object *object...*	Returns *true* if the hash includes a value for *key*
.value?	Object *object...*	Returns *true* if the collection includes a key for *value*
.index	Object *object...*	Returns the key for *value*
.invert		Returns a new *Hash* with keys and values inverted
.keys		Returns an *Array* with all keys from the *Hash*
.length		Returns the number of key-value pairs
.merge	Hash *hash...*	Returns a new *Hash* with entries from both Hashes
.select { |*object*| … }		Returns an *Array* with key-value pairs for which the block returns true
.to_a		Returns an *Array* with nested key-value pairs
.values		Returns an *Array* with all values from the *Hash*

String (Enumerable)

[] [] []	Fixnum *index* Range *range* Regexp *regexp*	Returns the specified character or *string*
.capitalize		Returns a capitalized version of the string
.center	Fixnum *width*, [String *filler*]	Centers the string using spaces or a specified filler string
.chomp	[String *separator*]	Returns a String with *separator* removed from the end
.count		Returns the number of characters
.downcase		Returns a lowercase version of the string
.gsub	Regexp *regexp* String *replacement*	Replaces all occurrences of *regexp* with replacement
.gsub { |*string...*| … }	Regexp *regexp*	Finds all occurrences of *regexp* and replaces them with the result of the block
.index	String/Regexp *piece*	Returns the position of the first occurrence of *piece*
.rindex	String/Regexp *piece*	Returns the position of the last occurrence of *piece*
.scan { |*string...*| … }	Regexp *regexp*	Scans the string for occurrences of *regexp* and passes them to the block
.split	String *string*	Splits the string into an array and returns it
.strip		Returns a string with whitespace removed from both ends
.swapcase		Returns a version of the string with uppercase turned to lowercase and vice-versa
.to_sym		Returns a symbol named like the string
.upcase		Returns an uppercase version of the string

IO

#read	String *filename* [, Fixnum *length*]	Opens *filename* and reads at most *length* bytes
#readline	String *file*, []	Reads and returns a line from *file*
.close		Closes the IO
.each_line { |*string*| … }		Send each line to the block
.eof?		Returns *true* if there is no more data to read
.print	Object *object*	Writes *object* to the IO
.printf	String *string*, [Object *object...*]	Formats and writes *string* to the IO
.puts	Object *object*	Writes *object* to the IO
.read	[Fixnum *length*]	Reads and returns at most *length* bytes
.readline		Reads and returns a line
.pos		Returns the current position in bytes

File < IO

#basename	String *path* [, String *suffix*]	Returns the filename from *path* with or without *suffix*
#exist?	String *filename*	Returns true if *filename* exists
#join	String *piece* [, String *piece*]	Returns path by joining *pieces*
#new { |file| … }	String *filename*, String *options*	Opens and sends *filename* to block
#new	String *filename*, String *options*	Opens and returns *filename*
#size	String *filename*	Returns the filesize of *filename*

File options

r/r+	Read/read-write from start of file
w/w+	Write/read-write truncate if file exists or create new
a/a+	Write/read-write from the end of the existing file or create new

Struct

.each { |*object*| … }		Calls the block for each instance variable passing the value
.each_pair { |*symbol, object*| … }		Calls the block for each instance variable passing the name and value
.length		Returns the number of instance variables
.members		Returns an *Array* containing all instance variable names
#new	[Symbol *name*, ...]	Creates a new *Struct* with an instance variable for each symbol

Kernel

block_given?		Returns *true* if a block was passed to the method
fork { … }		Creates a subprocess, runs the block in it and returns its ID
open	String *filename*	Opens a file
open { |io| … }	String *filename*	Opens a file, passes it to the block and closes it afterwards
p	Object *object*	Prints *object* to the stdio
printf	String *string*, [Object *object...*]	Formats and writes *string* to the stdio
lambda {|object...| … }		Creates and returns a new proc *object* with the supplied block
puts	String *string*	Writes *object* to the IO
require	String *filename*	Load a Ruby file
system(string [,string...])	String *command* [, *args*]	Executes a system command

RUBY 1.9

Ruby 1.9 is the new version of Ruby considered transitional to Ruby 2.0 containing many changes to the language and libraries. It has an entirely new virtual machine and bytecode compiler, formerly known as YARV.

The new version includes support for unicode in strings, the famous Oniguruma regular expression engine as well as Operating System Threads and Fibers for lightweight concurrency.

Important syntax additions/differences to Ruby 1.8

Syntax Additions/ Differences	Ruby 1.9	Ruby 1.8
Hash literal syntax	{ key: "value" }	{ :key => "value" }
Additional Proc/lambda definition syntax	foo = ->(a,b){ … }	foo = lambda { |a,b| … }
Additional Proc/lambda call syntax	foo.("x", "y")	foo.call("x", "y")
Block local variables	foo = lambda { |;a| … }	
Encoding support for String	"foo".encoding	
String indices return Strings	"foo"[2] # => 111	"foo"[2] # => "o"
Optional arguments are possible before and after other arguments	def foo(a, b = 2, c, d = 3) … end	
External Iterators	i = [1, 2, 3].each	

RESOURCES

The official Ruby website	http://www.ruby-lang.org
The official documentation	http://www.ruby-doc.org
The main Ruby repository	http://www.rubyforge.org
Wikipedia's overview of Ruby	http://en.wikipedia.org/wiki/Ruby_(programming_language)
The Ruby mailing lists	http://www.ruby-forum.com

Ruby Zone	http://ruby.dzone.com/
An interac tive online tutorial	http://tryruby.hobix.com (no download or installation)
A Ruby news site	http://www.rubyinside.com
A community-powered Ruby news site	http://www.rubyflow.com/
A Ruby-related blog aggregator	http://www.rubycorner.com
JRuby (Java Ruby Implementation)	http://jruby.codehaus.org
IronRuby (.NET Ruby Implementation)	http://www.ironruby.net

ABOUT THE AUTHORS

Melchior Brislinger

Melchior Brislinger is currently a student of Visual Communication at the Bauhaus-University Weimar, Germany. He uses Ruby and other programming languages and tools to explore the possibilities of combining art, design and technology.

Peter Cooper

Peter Cooper is a digital "jack of all trades" based in the north of England. He is author of *Beginning Ruby*—published by Apress— creator of numerous Web sites and technologies, a professional blogger who runs Ruby Inside—the most popular blog for Ruby and Rails developers—and an entrepreneur who sold two startups in 2007.

Publications
- *Beginning Ruby,* Apress

Projects
Ruby Inside—http://www.rubyinside.com/
Ruby Flow—http://www.rubyflow.com/
Rails Inside—http://www.railsinside.com/
SwitchPipe—http://switchpipe.org/

Blog
http://www.peterc.org/
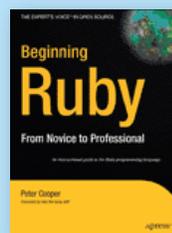http://twitter.com/peterc

Homepage
http://www.petercooper.co.uk/

RECOMMENDED BOOK

Beginning Ruby is for every type of reader wanting to learn Ruby, from novice programmers to web developers to Ruby newcomers. It starts by explaining the principles behind object-oriented programming, builds toward creating a genuine Ruby application, then explains key Ruby principles, such as classes and objects; projects, modules, and libraries; and other aspects of Ruby such as database access.

BUY NOW
books.dzone.com/books/beginning-ruby

Get More FREE Refcardz. Visit refcardz.com now!

Upcoming Refcardz:

Core Seam

Core CSS: Part III

Hibernate Search

Equinox

EMF

XML

JSP Expression Language

ALM Best Practices

HTML and XHTML

Available:

Essential MySQL

JUnit and EasyMock

Getting Started with MyEclipse

Spring Annotations

Core Java

Core CSS: Part II

PHP

Getting Started with JPA

JavaServer Faces

Core CSS: Part I

Struts2

Core .NET

Very First Steps in Flex

C#

Groovy

NetBeans IDE 6.1 Java Editor

RSS and Atom

GlassFish Application Server

Silverlight 2

IntelliJ IDEA

Visit **refcardz.com** for a complete listing of available Refcardz.

FREE
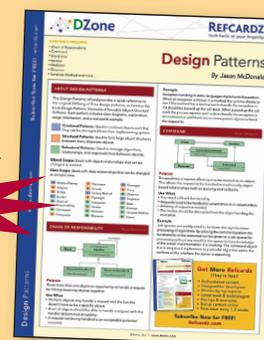
Design Patterns
Published June 2008

DZone

DZone communities deliver over 4 million pages each month to more than 1.7 million software developers, architects and decision makers. DZone offers something for everyone, including news, tutorials, cheatsheets, blogs, feature articles, source code and more. **"DZone is a developer's dream,"** says PC Magazine.

Version 1.0