Ruby 哈希（Hash） Ruby 范围（Range）

Ruby 日期 & 时间（Date & Time）

Time 类在 Ruby 中用于表示日期和时间。它是基于操作系统提供的系统日期和时间之上。该类可能无法表示 1970 年之前或者 2038 年之后的日期。

本教程将让您熟悉日期和时间的所有重要的概念。

创建当前的日期和时间

下面是获取当前的日期和时间的简单实例：

实例

```
#!/usr/bin/ruby -w
# -*- coding: UTF-8 -*-
time1 = Time.new
puts "当前时间 : " + time1.inspect
# Time.now 功能相同
time2 = Time.now
puts "当前时间 : " + time2.inspect
```

尝试一下 »

以上实例运行输出结果为：

```
当前时间 : 2015-09-17 15:23:14 +0800
当前时间 : 2015-09-17 15:23:14 +0800
```

获取 Date & Time 组件

我们可以使用 *Time* 对象来获取各种日期和时间的组件。请看下面的实例：

实例

```
#!/usr/bin/ruby -w
# -*- coding: UTF-8 -*-
time = Time.new
# Time 的组件
puts "当前时间 : " + time.inspect
puts time.year # => 日期的年份
puts time.month # => 日期的月份（1 到 12）
puts time.day # => 一个月中的第几天（1 到 31）
puts time.wday # => 一周中的星期几（0 是星期日）
puts time.yday # => 365：一年中的第几天
puts time.hour # => 23：24 小时制
puts time.min # => 59
puts time.sec # => 59
```

```
puts time.usec # => 999999: 微秒
puts time.zone # => "UTC": 时区名称
```

以上实例运行输出结果为：

```
当前时间 : 2015-09-17 15:24:44 +0800
2015
9
17
4
260
15
24
44
921519
CST
```

Time.utc、Time.gm 和 Time.local 函数

这些函数可用于格式化标准格式的日期，如下所示：

```
# July 8, 2008
Time.local(2008, 7, 8)
# July 8, 2008, 09:10am, 本地时间
Time.local(2008, 7, 8, 9, 10)
# July 8, 2008, 09:10 UTC
Time.utc(2008, 7, 8, 9, 10)
# July 8, 2008, 09:10:11 GMT (与 UTC 相同)
Time.gm(2008, 7, 8, 9, 10, 11)
```

下面的实例在数组中获取所有的组件：

```
[sec,min,hour,day,month,year,wday,yday,isdst,zone]
```

尝试下面的实例：

实例

```
#!/usr/bin/ruby -w
time = Time.new
values = time.to_a
p values
```

以上实例运行输出结果为：

```
[39, 25, 15, 17, 9, 2015, 4, 260, false, "CST"]
```

该数组可被传到 *Time.utc* 或 *Time.local* 函数来获取日期的不同格式，如下所示：

实例

```
#!/usr/bin/ruby -w
time = Time.new
values = time.to_a
puts Time.utc(*values)
```

以上实例运行输出结果为：

```
2015-09-17 15:26:09 UTC
```

下面是获取时间的方式，从纪元以来的秒数（平台相关）：

```
# 返回从纪元以来的秒数
time = Time.now.to_i
# 把秒数转换为 Time 对象
Time.at(time)
# 返回从纪元以来的秒数，包含微妙
time = Time.now.to_f
```

时区和夏令时

您可以使用 *Time* 对象来获取与时区和夏令时有关的所有信息，如下所示：

```
time = Time.new
# 这里是解释
time.zone # => "UTC": 返回时区
time.utc_offset # => 0: UTC 是相对于 UTC 的 0 秒偏移
time.zone # => "PST"（或其他时区）
time.isdst # => false: 如果 UTC 没有 DST（夏令时）
time.utc? # => true: 如果在 UTC 时区
time.localtime # 转换为本地时区
time.gmtime # 转换回 UTC
time.getlocal # 返回本地区中的一个新的 Time 对象
time.getutc # 返回 UTC 中的一个新的 Time 对象
```

格式化时间和日期

有多种方式格式化日期和时间。下面的实例演示了其中一部分：

实例

```
#!/usr/bin/ruby -w
time = Time.new
puts time.to_s
puts time.ctime
puts time.localtime
puts time.strftime("%Y-%m-%d %H:%M:%S")
```

以上实例运行输出结果为：

```
2015-09-17 15:26:42 +0800
Thu Sep 17 15:26:42 2015
```

```
2015-09-17 15:26:42 +0800
2015-09-17 15:26:42
```

时间格式化指令

下表所列出的指令与方法 *Time.strftime* 一起使用。

指令	描述
%a	星期几名称的缩写（比如 Sun）。
%A	星期几名称的全称（比如 Sunday）。
%b	月份名称的缩写（比如 Jan）。
%B	月份名称的全称（比如 January）。
%c	优选的本地日期和时间表示法。
%d	一个月中的第几天（01 到 31）。
%H	一天中的第几小时，24 小时制（00 到 23）。
%I	一天中的第几小时，12 小时制（01 到 12）。
%j	一年中的第几天（001 到 366）。
%m	一年中的第几月（01 到 12）。
%M	小时中的第几分钟（00 到 59）。
%p	子午线指示（AM 或 PM）。
%S	分钟中的第几秒（00 或 60）。
%U	当前年中的周数，从第一个星期日（作为第一周的第一天）开始（00 到 53）。
%W	当前年中的周数，从第一个星期一（作为第一周的第一天）开始（00 到 53）。
%w	一星期中的第几天（Sunday 是 0，0 到 6）。
%x	只有日期没有时间的优先表示法。
%X	只有时间没有日期的优先表示法。
%y	不带世纪的年份表示（00 到 99）。
%Y	带有世纪的年份。

%Z	时区名称。
%%	% 字符。

时间算法

您可以用时间做一些简单的算术，如下所示：

```
now = Time.now # 当前时间
puts now
past = now - 10 # 10 秒之前。Time - number => Time
puts past
future = now + 10 # 从现在开始 10 秒之后。Time + number => Time
puts future
diff = future - now # => 10 Time - Time => 秒数
puts diff
```

以上实例运行输出结果为：

```
2015-09-17 15:27:08 +0800
2015-09-17 15:26:58 +0800
2015-09-17 15:27:18 +0800
10.0
```

← Ruby 哈希（Hash）

Ruby 范围（Range）→