开始

模块属性

Elixir 中的模块属性有三个用途：

1. 它们用于注释模块，通常包含用户或 VM 要使用的信息。
2. 它们作为常量工作。
3. 它们用作编译期间使用的临时模块存储。

让我们逐一检查每个案例。

作为注释

Elixir 从 Erlang 带来了模块属性的概念。例如：

```
defmodule MyServer do
  @moduledoc "My server code."
end
```

在上面的示例中，我们使用模块属性语法定义模块文档。长生不老药有一些保留属性。以下是其中的一些，最常用的：

- `@moduledoc` - 提供当前模块的文档。
- `@doc` - 为属性后面的函数或宏提供文档。
- `@spec` - 为属性后面的函数提供类型规范。

新闻：Elixir v1.15 发布

搜索。。。

接口文档

开始

1. 介绍
2. 基本类型
3. 基本运算符
4. 模式匹配
5. 案例、cond 和 if
6. 二进制文件、字符串和字符列表
7. 关键字列表和地图
8. 模块和功能
9. 递归
10. 枚举项和流
11. 过程
12. IO 和文件系统
13. 别名、要求和导入
14. 模块属性
15. 结构体
16. 协议
17. 理解
18. 印记
19. 尝试、捕捉和救援
20. 可选语法表
21. Erlang 库

- `@behaviour` -（注意英式拼写）用于指定 OTP 或用户定义的行为。

`@moduledoc` 并且是迄今为止最常用的属性，我们希望您经常使用它们。Elixir 将文档视为一流，并提供许多访问文档的功能。您可以在我们的官方文档中阅读有关在 Elixir 中编写文档的更多信息。`@doc`

让我们回到前面章节中定义的模块，添加一些文档并将其保存到文件中：`Math` `math.ex`

```
defmodule Math do
  @moduledoc """
  Provides math-related functions.

  ## Examples

      iex> Math.sum(1, 2)
      3

  """

  @doc """
  Calculates the sum of two numbers.
  """
  def sum(a, b), do: a + b
end
```

Elixir 提倡使用 Markdown 和 heredocs 来编写可读的文档。Heredocs 是多行字符串，它们以三重双引号开头和结尾，保持内部文本的格式。我们可以直接从 IEx 访问任何编译模块的文档：

```
$ elixirc math.ex
$ iex
```

```
iex> h Math # Access the docs for the module Math
...
iex> h Math.sum # Access the docs for the sum function
...
```

我们还提供了一个名为 ExDoc 的工具，用于从文档生成 HTML 页面。

您可以查看模块的文档，了解受支持属性的完整列表。Elixir 还使用属性来定义类型规范。

本节介绍内置属性。但是，开发人员也可以使用属性或库扩展属性以支持自定义行为。

作为“常量”

Elixir 开发人员在希望使值更可见或可重用时经常使用模块属性：

```
defmodule MyServer do
  @initial_state %{host: "127.0.0.1", port: 3456}
  IO.inspect @initial_state
end
```

尝试访问未定义的属性将打印警告：

```
defmodule MyServer do
  @unknown
end
warning: undefined module attribute @unknown, please
remove access to @unknown or explicitly set it before
access
```

属性也可以在函数中读取：

```
defmodule MyServer do
  @my_data 14
  def first_data, do: @my_data
  @my_data 13
  def second_data, do: @my_data
end

MyServer.first_data #=> 14
MyServer.second_data #=> 13
```

> *注意：不要在属性和它的值之间添加换行符，否则 Elixir 会认为你正在读取值，而不是设置它。*

定义模块属性时可以调用函数：

```
defmodule MyApp.Status do
  @service URI.parse("https://example.com")
  def status(email) do
    SomeHttpClient.get(@service)
  end
end
```

上面的函数将在编译时调用，它的*返回值*，而不是函数调用本身，将被替换为属性。所以上面将有效地编译成这个：

```
defmodule MyApp.Status do
  def status(email) do
    SomeHttpClient.get(%URI{
      authority: "example.com",
      host: "example.com",
      port: 443,
      scheme: "https"
    })
  end
end
```

这对于预计算常量值可能很有用，但如果希望在运行时调用函数，它也可能会导致问题。例如，如果要从数据库或属性中的环境变量中读取值，请注意，它只会在编译时读取该值。但是要小心：*不能调用与属性本身相同的模块中定义的函数*，因为在定义属性时尚未编译它们。

每次在函数中读取属性时，Elixir 都会拍摄其当前值的快照。因此，如果您在多个函数中多次读取同一属性，则最终可能会创建它的多个副本。这通常不是问题，但如果使用函数来计算大型模块属性，则可能会减慢编译速度。解决方案是将属性移动到共享函数。例如，代替这个：

```
def some_function, do: do_something_with(@example)
def another_function, do: do_something_else_with(@example)
```

更喜欢这个：

```
def some_function, do: do_something_with(example())
def another_function, do:
do_something_else_with(example())
defp example, do: @example
```

如果计算成本低，则最好完全跳过模块属性，并在函数中计算其值。
`@example`

累积属性

通常，重复模块属性会导致重新分配其值，但在某些情况下，您可能需要配置模块属性以便累积其值：

```
defmodule Foo do
  Module.register_attribute __MODULE__, :param,
accumulate: true

  @param :foo
  @param :bar
  # here @param == [:bar, :foo]
end
```

作为临时存储

要查看使用模块属性作为存储的示例，只需看看Elixir的单元测试框架ExUnit即可。ExUnit 将模块属性用于多种不同目的：

```
defmodule MyTest do
  use ExUnit.Case, async: true

  @tag :external
  @tag os: :unix
  test "contacts external service" do
    # ...
  end
end
```

在上面的示例中，将 的值存储在模块属性中，以更改模块的编译方式。标记也定义为属性，它们存储可用于设置和筛选测试的标记。例如，可以避免在计算机上运行外部测试，因为它们速度很慢且依赖于其他服务，而仍

然可以在生成系统中启用它们。`ExUnit` `async: true` `accumulate: true`

为了理解底层代码，我们需要宏，因此我们将在元编程指南中重新访问此模式，并学习如何使用模块属性作为存储，以允许开发人员创建域特定语言（DSL）。

在接下来的章节中，我们将探讨结构和协议，然后再讨论异常处理和其他结构，如符号和推导。

← 上一页　返回页首　下一→