Elixir-Language

http://elixir-lang.org/

内容大纲

- 语言概述
- 基本类型
- 语句
- 通信机制
- socket示例

两句语概括

- 一种函数式编程语言
- 基于Erlang的虚拟机

语言历史

- Beginning of 2011
- April 2011 v0.3.0 stable

  Plataformatec

- 2012-05-25 v0.5.0 stable

  1.完全与Erlang兼容

  2.github语法高亮支持

  3.库的标准化

- 当前版本：v1.1.1 (2015-09-29)

José Valim

语言历史

- Beginning of 2011
- April 2011 v0.3.0 stable

  Plataformatec

- 2012-05-25 v0.5.0 stable

  1.完全与Erlang兼容

  2.github语法高亮支持

  3.库的标准化

- 当前版本：v1.1.1 (2015-09-29)

José Valim

设计目标

- 兼容性(Compatibility)
- 高效(Productivity)
- 语法、可读、简洁
- 可扩展性（宏的支持、DSL）

参考:http://elixir-lang.org/blog/2013/08/08/elixir-design-goals/

基本数据类型

- Booleans
- Atoms
- Strings
- Anonymous functions
- Lists
- Tuples
- Strings

Booleans & Atoms

- Booleans

  true/false is_boolean/1

- Atoms 常量

  :hello

Anonymous functions

- 匿名函数 (Lambda表达式)

```
(function(){console.log("hello world!”)})()
```

```
add = fn a, b -> a + b end

add.(1,2)
```

另一种写法： `(fn a, b -> a + b end).(1,2)`

- is_function/1

List & Tuples & Strings

- (Linked) List (列表)

  [1, 2, true, 3]

- Tuples (元组、数组)

  {:ok, “hello"}

- String

  string = "hełło"

复杂数据类型

- Keyword lists

```
list = [{:a, 1}, {:b, 2}]
```

```
list = [a: 1, b: 2]
```

- Maps

```
map = %{:a => 1, 2 => :b}
```

```
map[:a]  map[2]  (map.a)
```

Modules模块化

```
defmodule Concat do
  def join(a, b \\ nil, sep \\ " ")
  def join(a, b, _sep) when is_nil(b) do
    a
  end
  def join(a, b, sep) do
    a <> sep <> b
  end
  defp hi() do
    IO.puts "private use only"
  end
end
```

concat.ex

条件判断

- case
- cond
- if else / unless (宏)

condition.ex

Enumerables & Pipe

- 主要通过Enum提供的方法

1. map
2. reduce
3. filter
4. sum

- 管道操作

demoenum.ex

IO & 文件操作

- IO

  IO.puts "hello world”

  IO.gets "yes or no? "

- File

demofile.ex

Structs

- defstruct

```
defmodule User do

  defstruct name: "john", age: 21

end
```

- a = %User{}
- b = %User{name: "点评", age: 12}
- b = %{b | name: "新美大"}

Protocols 协议

- 协议 — 与java里的接口类似
- defprotocol/defimpl

```
defprotocol Blank do
  @doc "Returns true if data is considered blank”
  def blank?(data)
end
```

protocol.ex

异常处理

- try/rescue
- raise
- 自定义异常

```
defmodule MyError do
  defexception message: "default message"
end
raise MyError
```

杂项

- alias, require
- module attributes

@moduledoc

@doc

@xxx　　# 作为常量用

Process

- “进程” lightweight process
- 创建process: spawn/1

```
pid = spawn fn -> 1 + 2 end
```

- Process模块

进程间通信

- 通过消息 send/receive
- 见 demo

client.ex/server.ex

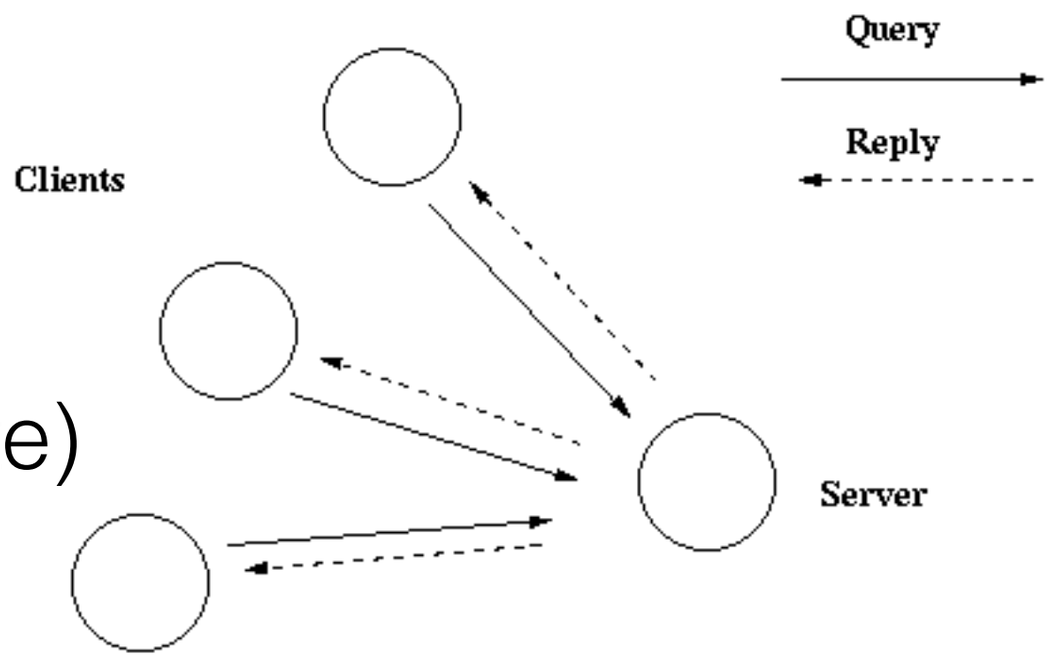
通用方式

- GenServer
- GenEvent
- Task

genclient.ex/genserver.ex

GenServer机制

- Client-Server Model
- hand_call(request, from, state)
- hand_cast(request, state)
- hand_info(msg, state)



The Client–server model

一个socket的例子

- demo

socket_demo_server.ex

工具介绍

- 构建工具mix

  mix compile / mix run —no-halt

- 测试框架ExUnit

- 文档工具ex_doc

  mix docs

Phoenix Framework

- Elixir 的MVC框架

- 类似的有sugar-framework

语言生态对比

	Elixir	Java
构建工具	mix	maven
仓库	hex	maven仓库
测试 工具	ExUnit	JUnit
数据库中间件	Ecto	ibatis/mybatis
MVC框架	Phoenix/Sugar	SpringMVC
开发工具	emacs/vim/sublime/atom	Eclipse/Intellij
文档	ex_doc	java doc

参考链接

- http://www.creativedeletion.com/2015/04/19/elixir_next_language.html
- http://elixir-lang.org/
- http://elixir-lang.org/getting-started/introduction.html
- http://www.erlang.org/doc/design_principles/gen_server_concepts.html
- https://github.com/elixir-lang