Erlang Academy

Лекция 3

Автор курса: Костюшкин Сергей

План

- proplists and map
- Охранные выражения (guards)
- Операторы case, if, блок begin..end
- Исключения
- Немного о типизации
- Конвертация типов
- Обработчики списков
- Бинарные строки и битовый синтаксис
- Обработчики бинарных данных

proplists

```
1> Proplist = [{name, "Santa"}, {age, 1054}].
[{name,"Santa"},{age,1054}]
2> proplists:get_value(name, Proplist).
"Santa"
```

maps

```
1> Map = #{name => "Santa", age => 1054}.
#{age => 1054,name => "Santa"}
2> Map2 = maps:put(sex, male, Map).
#{age => 1054,name => "Santa",sex => male}
3> maps:get(name, Map2).
"Santa"
4> #{sex := Sex} = Map2.
#{age => 1054,name => "Santa",sex => male}
5> Sex.
male
```

Охранные выражения (Guards)

```
get_user_status({user, _Name, Gender, Age}) when Gender =:= female, Age < 21 ->
    girl;
get_user_status({user, _Name, Gender, Age}) when Gender =:= female, Age >= 21 ->
    women;
get_user_status({user, _Name, Gender, Age}) when Gender =:= male, Age < 21 ->
    boy;
get_user_status({user, _Name, Gender, Age}) when Gender =:= male, Age >= 21 ->
    men.
```

Функции охранники

is_atom/1
is_binary/1
is_bitstring/1
is_boolean/1
is_builtin/3
is_float/1
is_function/1
is_function/2
is_integer/1

is_list/1
is_number/1
is_pid/1
is_port/1
is_record/2
is_record/3
is_reference/1
is_tuple/1

Оператор case

```
case Expr of
    Pattern Guards -> …
    Pattern Guards -> ...
    Pattern Guards -> …
    _ -> ...
end.
```

Оператор case

```
insert(X,[]) ->
    [X];
insert(X,Set) ->
    case lists:member(X,Set) of
        true  -> Set;
        false -> [X|Set]
    end.
```

Оператор if

```
if
    BooleanExpr -> ….
    BooleanExpr -> ….
    BooleanExpr -> ….
    true        ->
end.
```

Оператор if

```
help_me(Animal) ->
    if
        Animal == cat    -> "meow";
        Animal == cow  -> "mooo";
        Animal == dog  -> "woof";
        true -> "fgdadfgna"
    end.
```

Обработка исключений

```
X = try Expression
catch
    error:ExceptionPattern -> Expression2;
    exit:ExceptionPattern -> Expression2;
    throw:ExceptionPattern -> Expression2;
    ExceptionPattern -> … %% Аналогично throw
after %% Эта часть будет выполнятся всегда
    Expression3 %% Ошибка здесь ни на что не повлияет
end
```

Обработка исключений

```
X = try Expression of
    SuccessfulPattern [Guards] -> Expression1
catch
    error:ExceptionPattern -> Expression2;
    exit:ExceptionPattern -> Expression2;
    throw:ExceptionPattern -> Expression2;
    ExceptionPattern -> … %% Аналогично throw
after %% Эта часть будет выполнятся всегда
    Expression3 %% Ошибка здесь ни на что не повлияет
end
```

Обработка исключений (Erlang 21)

```
X = try Expression of
    SuccessfulPattern [Guards] -> Expression1
catch
    error:ExceptionPattern:StackTrace -> Expression2;
    exit:ExceptionPattern:StackTrace -> Expression2;
    throw:ExceptionPattern:StackTrace -> Expression2;
    ExceptionPattern -> … %% Аналогично throw
after %% Эта часть будет выполнятся всегда
    Expression3 %% Ошибка здесь ни на что не повлияет
end
```

Обработка исключений

```
case catch Expression of
    SuccessfulPattern [Guards] -> Expression1
    {'EXIT', ExceptionPattern} -> Expression2
end.
```

Обработка исключений

```
1> catch throw(whoa).
whoa
2> catch exit(die).
{'EXIT',die}
3> catch 1/0.
{'EXIT',{badarith,[{erlang,'/',[1,0]},
                  {erl_eval,do_apply,5},
                  {erl_eval,expr,5}, {shell,exprs,6},
                  {shell,eval_exprs,6}, {shell,eval_loop,3}]}}
4> catch 2+2.
4
```

Конвертация типов

atom_to_binary/2
atom_to_list/1
binary_to_atom/2
binary_to_existing_atom/2
binary_to_list/1
bitstring_to_list/1
binary_to_term/1
float_to_list/1
fun_to_list/1
integer_to_list/1
integer_to_list/2
iolist_to_binary/1
iolist_to_atom/1
list_to_atom/1
list_to_binary/1
list_to_bitstring/1
list_to_existing_atom/1
list_to_float/1
list_to_integer/2
list_to_pid/1
list_to_tuple/1
pid_to_list/1
port_to_list/1
ref_to_list/1
term_to_binary/1
term_to_binary/2
tuple_to_list/1

Обработчики списков

```
[X + 1 || X <- [1,2,3,4,5,6] ].
[X || X <- [1,2,a,3,4,b,5,6], X > 3].
[X || X <- [1,2,a,3,4,b,5,6], is_integer(X), X > 3].
[X || X <- [1,2,3,4,5,6,7], X rem 2 =:= 0].
[{X, Y} || X <- [1,2,3], Y <- [a,b]].
[Y || {_X, Y} <- L].
[begin
    X1 = binary_to_integer(X),
    Y1 = binary_to_integer(Y),
    X1+Y1
end || {X,Y} <- L, is_binary(X), is_binary(Y)].
```

Бинарные данные

```
Bin1 = <<1,2,3,0,255>>.
Bin2 = <<"Some Text">>.
Bin2 = <<83, 111, 109, 101, 32, 84, 101, 120, 116>>.

<<"So", X, Rest/binary>> = Bin2.
%% X = 109, Rest = <<101, 32,84, 101, 120, 116>>
<<"So", Y:16, Rest2/binary>> = Bin2.
%% Y = 28005, Rest2 = <<32,84, 101, 120, 116>>
<<"So", Y:16/integer, Rest2/binary>> = Bin2.
%% Y = 28005, Rest2 = <<32,84, 101, 120, 116>>
```

Смешаные обработчики

```
Bin = <<1,2,3>>.
List = [1,2,3].

<< <<(X+1)>> || <<X>> <= Bin >>.
<< <<(X+1)/utf8>> || <<X/utf8>> <= Bin >>.
[X+1 || <<X>> <= Bin].
[X+1 || X <- List].
<< <<(X+1)>> || X <- List>>.
```

Для домашнего чтения

Bit Syntax Guide