Erlang Academy

Лекция 5

Автор курса: Костюшкин Сергей

План

- Метапрограммирование
- Параметризированые модули
- Списки свойств (модули lists и proplists)
- Записи (records)
- ETS
- DETS
- MatchSpec - Язык запросов в ETS и DETS
- Фреймворк для тестирования Eunit

Метапрограммирование

Парадигма программирования при которой программы трактуются как данные. Это означает что программа может читать, гененерировать и изменять другие программы или даже себя.

Виды метапрограммирования:

- Кодогенерация
- Статический анализ кода
- DSL (Domain Specific language) — предметно-специфичный язык

Параметризированые модули

Модуль:

```
-module(param_example, [Name]).
-export([name/0]).

name() -> Name.
```

Консоль:

```
1> Obj = param_example:new("Tony").
2> Obj:name().
"Tony"
```

proplists

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

proplists

```
PropList = [{key1, val1}, {key2, val2}].
A = proplists:get_value(key1, PropList). %% A = val1
B = proplists:get_value(key3, PropList, foo). %% B = foo


%% Faster alternative to proplists:get_value/3 from cowboy
get_value(Key, Opts, Default) ->
    case lists:keyfind(Key, 1, Opts) of
        {_, Value} -> Value;
        _ -> Default
    end.
```

maps

```
1> Map = #{name => "Santa", age => 1054}.
#{age => 1054,name => "Santa"}
2> maps:get(name, Map).
"Santa"
3> #{age := Age} = Map.
#{age => 1054,name => "Santa"}
4> Age.
1054
5> Map2 = Map#{age => 2000}.
#{age => 2000,name => "Santa"}
```

Process Dictionary

```
1> put(key1, value1).
undefined
2> put(key2, value2).
undefined
3> put(key3, value3).
undefined
4> get().
[{key3,value3},{key2,value2},{key1,value1}]
5> get(key2).
value2
6> get_keys().
[key3,key2,key1]
7> erase().
[{key3,value3},{key2,value2},{key1,value1}]
```

dict

```
1> D = dict:new().
{dict,0,16,16,8,80,48,
      {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},
      {{[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]}}}
2> D2 = dict:append(key1, value1, D).
{dict,1,16,16,8,80,48,
      {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},
      {{[],
        [[key1,value1]],
        [],[],[],[],[],[],[],[],[],[],[],[],[],[]}}}
3> dict:size(D2).
1
4> dict:erase(key1, D2).
{dict,0,16,16,8,80,48,
      {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]},
      {{[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],[]}}}
```

Записи (records)

```
-module(records_demo).
-export([new/0, new/3]).
-export([get_field/2, set_filed/3, get_index/1]).

-record(person, {name=”Joe”, gender, age=56}).

new() ->
    #person{}.

new(Name, Gender, Age) ->
    #person{name=Name, gender=Gender, age=Age}.
```

Записи (records) продолжение

```
get_index(name)   -> #person.name;
get_index(gender) -> #person.gender;
get_index(age)    -> #person.age.

get_field(name, Record)   -> Record#person.name;
get_field(gender, Record) -> Record#person.gender;
get_field(age, Record)    -> Record#person.age.

set_filed(name, Name, Record) ->
    Record#person{name=Name}.
set_filed(age, Age, Record) ->
    Record#person{age=Age}.
```

ETS

```
EtsId = ets:new(TableName, Opts).
true = ets:insert(EtsId, ObjectOrObjects).
[Object] = ets:lookup(EtsId, Key).
true = ets:delete(EtsId).
```

```
persons = ets:new(persons, [public, named_table]).
true = ets:insert(persons, {key1, value1}).
[{key1, value1}] = ets:lookup(persons, key1).
true = ets:delete(persons).
```

DETS

```
{ok, Name} = dets:open_file(Name, Opts).
ok = dets:insert(Name, ObjectOrObjects).
[Object] = dets:lookup(Name, Key).
ok = dets:close(Name).
```

```
{ok, persons} = dets:open_file(persons, []).
ok = dets:insert(persons, {key1, value1}).
[{key1,value1}] = dets:lookup(persons, key1).
ok = dets:close(persons).
```

MatchSpec

[Match] = ets:select(EtsId, MatchSpec).

ets:select(persons,[{{'$1','$2','$3'},[],['$$']}]).