

Erlang - funkcionalno rešenje za konkurentni svet

Seminarski rad u okviru kursa
Metodologija strunog i naunog rada
Matematiki fakultet

Tijana Jevti, Jelena Mrdak, David Dimi, Zorana Gaji
tijanatijanajevtic@gmail.com, mrdakj@gmail.com,
daviddimic@hotmail.com, zokaaa_gajich@bk.ru

6. april 2019.

Sažetak

U ovom radu je prikazan programski jezik Erlang iz razliitih uglova. Kroz niz poglavlja i primera, ispriana je njegova istorija - kad, kako, gde i zato je nastao, po emu je karakteristian, ta ga to izdvaja od drugih programskih jezika, koji su to koncepti koji su svojevrsni Erlangu. Nakon itanja rada, italac e imati globalnu sliku o jeziku i detaljniji pogled na neke vane koncepte, kao i uvid u korienu literaturu koju moe konsultovati radi daljeg informisanja o temi.

Sadržaj

1	Uvod	2
2	Nastanak i istorijski razvoj	2
3	Osnovna namena, svrha i mogunosti	3
4	Osnovne osobine	3
5	Okruenja (framework) i njihove karakteristike	3
6	Instalacija i pokretanje	3
7	Primeri kodova sa objanjenjima	3
8	Specifinosti	4
9	Zakljuak	4
	Literatura	4
A	Dodatak	4

1 Uvod

Primer 1.1 I tabele treba da budu u svom okruenju, i na njih je neopodno referisati se u tekstu. Na primer, u tabeli 1 su prikazana razliita poravnanja u tabelama.

Tabela 1: Razlita poravnanja u okviru iste tabele ne treba koristiti jer su nepregledna.

centralno poravnanje	levo poravnanje	desno poravnanje
a	b	c
d	e	f

2 Nastanak i istorijski razvoj

1981. godine je oformljena nova laboratorija, Erikson CSLab (eng. The Ericsson CSLab) u okviru firme Erikson sa ciljem da predlae i stvara nove arhitekture, koncepte i strukture za budue softverske sisteme. Eksperimentisanje sa dodavanjem konkurentnih procesa u programski jezik Prolog je bio jedan od projekata Erikson CSLab-a i predstavlja zaetak novog programskog jezika. Taj programski jezik je 1987. godine nazvan Erlang¹. Sve do 1990., Erlang se mogao posmatrati kao dijalekt Prologa. Od tada, Erlang ima svoju sintaksu i postoji kao potpuno samostalan programski jezik. Godine rada su rezultirale u sve brim, boljim i stabilnijim verzijama jezika, kao i u nastanku standardne biblioteke OTP (eng. The Open Telecom Platform) [?]. Od decembra 1998. godine, Erlang i OTP su postali deo slobodnog softvera (eng. open source software) i mogu se slobodno preuzeti sa Erlangovog zvaninog sajta [?]. Danas, veliki broj kompanija koristi Erlang u razvoju svojih softverskih reenja. Neke od njih su: Erikson, Motorola, Votsap (eng. Whatsapp), Jahu (eng. Yahoo!), Amazon, Fejsbuk (eng. Facebook).

2.1 Uticaji drugih programskih jezika

Erlang je funkcionalan i konkurentan programski jezik. Na njega, kao na funkcionalan jezik, uticao je Lisp funkcionalnom paradigmom koju je prvi predstavio. Na planu konkurentnosti Erlang svojevrsan primer (detaljnije u odeljku 4).

Na poetku, Erlang je stvaran kao neki dodatak na Prolog, vremenom prerastao u dijalekt Prologa, a kasnije je zbog svoje kompleksnosti i sveobuhvatnosti evoluirao u potpuno novi programski jezik. Stoga je uticaj Prologa na Erlang bio neminovan. Sintaksa Erlanga u velikoj meri podsea na Prologovu (npr. promenljive moraju poinjati velikim slovom u oba jezika, svaka funkcionalna celina se zavrava takom), oba jezika u velikoj meri koriste poklapanje obrazaca (eng. pattern matching).

Sa druge strane, Erlang je uticao na nastanak programskog jezika Elixir (eng. Elixir).

¹Erlang je jedinica saobraaja u oblasti telekomunikacija i predstavlja kontinuirano korenje jednog kanala (npr. ako jedna osoba obavi jedan poziv telefonom u trajanju od sat vremena, tada se kae da sistem ima 1 Erlang saobraaja na tom kanalu).

3 Osnovna namena, svrha i mogućnosti

4 Osnovne osobine

4.1 Podrane paradigme

4.2 Koncepti

5 Okruženja (framework) i njihove karakteristike

6 Instalacija i pokretanje

Postoji vie naina da se instalira Erlang sa neophodnim paketima. U ovom odeljku e biti predstavljena instalacija korienjem prekompajliranih binarnih fajlova za neke operativne sisteme zasnovane na Linuksovom kernelu i pokretanje na jednom od njih, kao i instalacija za Windows.

6.1 Linux

Na operativnim sistemima zasnovanim na Ubuntu, Erlang se moe instalirati sa: `sudo apt-get install erlang`.

Nakon uspone instalacije, Erlang kod je mogue kompajlovati ili interpretirati i pokretati u interpretatoru. Interpretator se pokree kucanjem komande `erl` u terminalu, a iz istog se izlazi sa `Ctrl+G` iza kog sledi `q` [?]. Erlang interpretator ima u sebi ugrađen editor teksta koji je baziran na emacs-u [?].

Kod iz datoteke se kompajluje komandom `erlc` i navoenjem imena fajla sa ekstenzijom `erl`. Nakon toga se dobija izvorna datoteka sa ekstenzijom `beam` koja se moe pokrenuti uz navoenje adekvatnih flegova.

6.2 Windows

7 Primeri kodova sa objanjenjima

Poćemo od primera "Hello World". Da bismo željeni tekst prikazali u konzoli, korićemo `io` modul. Pritom, `~n` koristimo za novi red.

```
-module(hello_world).  
-compile(export_all).
```

```
hello() ->  
io:format("hello world~n").
```

Kao i većina funkcionalnih jezika, i Erlang podržava shvatanje listi (eng. list comprehensions), što ilustrujemo narednim primerima.

```
> [X || X <- [1,2,a,3,4,b,5,6], X > 3].  
[a,4,b,5,6]
```

Notacija `X <- [1, 2, a, ...]` je generator, dok je izraz `X>3` filter.

Možemo primeniti više filtera.

```
> [X || X <- [1,2,a,3,4,b,5,6], integer(X), X > 3].  
[4,5,6]
```

Takođe, moguće je kombinovati i generatore. Na primer, Dekartov proizvod dve liste možemo napisati kao

```
> [{X, Y} || X <- [1,2,3], Y <- [a,b]].
[{1,a},{1,b},{2,a},{2,b},{3,a},{3,b}]
```

Algoritam QuickSort u Erlangu se može implementirati na sledeći način:

```
sort([Pivot|T]) ->
  sort([ X || X <- T, X < Pivot]) ++
  [Pivot] ++
  sort([ X || X <- T, X >= Pivot]);
sort([]) -> [].
```

Izraz `[X || X <- T, X < Pivot]` je lista svih elemenata iz T koji su manji od pivota. Slično, `[X || X <- T, X >= Pivot]` je lista svih elemenata iz T koji su veći ili jednaki od pivota.

Neizostavna funkcija svih funkcionalnih programskih jezika jeste map. `map(F, List)` je funkcija koja prima funkciju F i listu L i vraća novu listu dobijenu primenom funkcije F na svaki element liste L.

```
map(F, [H|T]) -> [F(H)|map(F, T)];
map(F, []) -> [].

double(L) -> map(fun(X) -> 2*X end, L).
> double([1,2,3,4]).
[2,4,6,8]
```

8 Specifinosti

9 Zaključak

A Dodatak