Razvoj programskog jezika Scala

Seminarski rad u okviru kursa
Dizajn programskih jezika
Matematički fakultet

Nikolina Kuprešanin
avadak@yahoo.com

18. decembar 2019.

Sažetak

U ovom radu prikazuju se osnove razvoja programskog jezika Scala. Ukratko su opisani programski jezici koji su imali najveći uticaj na njegov razvoj: Java, OCaml i Erlang. Za svaki jezik date su osnovne informacije i prikazano je razvojno stablo koje uključuje ove jezike.

Sadržaj

1 Uvod

Razvoj programskih jezika podstaknut je razvojem tehnologija i težnjom da se prevaziđu problemi koji sa tim razvojem dolaze. Cilj je da se jezik prilagodi našim potrebama. Programski jezik *Scala*, kao što mu i ime kaže, se "skalira", raste sa potrebama programera. Može se koristiti za pisanje malih skripti ali i složenih softverskih sistema. [6] Nastaje 2003.godine i objedinjuje ideje objektno-orijentisane, funkcionalne i imperativne paradigme što doprinosi raznolikosti domena njegove upotrebe. Tvorac jezika je Martin Odersky, nemački naučnik čiji se rad zasniva na objektno-orijentisanoj i funkcionalnoj paradigmi. On smatra da su ove dve paradigme dve strane istog novčica koje treba ujediniti što je više moguce. Tako nastaje Scala, programski jezik opšte namene koji ih obe podržava.
Logo jezika Scala prikazan je na slici 1.

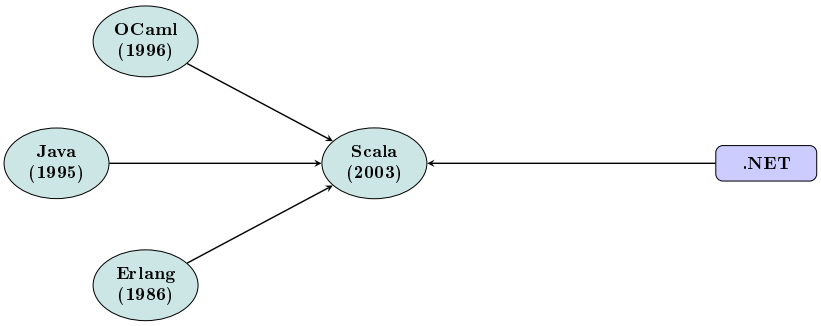
Slika 1: Logo programskog jezika Scala

2 Osnovno o programskom jeziku Scala

Scala je objektno-orijentisan, funkcionalan, statički tipiziran jezik adaptiran za paralelno izvršavanje i simbolički račun.[9] Javno je objavljena na Java platformi. Izvršni kod Scala programa se prevodi u Java bajt kod koji se izvršava na Java virtuelnoj mašini. Za razliku od Jave, Scala je čisto objektno orijentisan jezik- svaka vrednost je objekat. Posebno interesantan novitet u programskom jeziku Scala su trejtovi (engl. *traits*) koje su slični interfejsima u Javi, ali u njima mozemo definisati polja i metode. Koriste se za deljenje polja i interfejsa izmedju klasa.
Scala je i funkcionalan jezik pa se funkcije kreiraju i prosleđuju bez ikakvih restrikcija kao kod imperativnih jezika.[4] U Scali se srećemo sa pojmovima funkcija višeg reda, anonimnih, ugnježdnenih funkcija. Javlja se uparivanje obrazaca (engl. *pattern matching*) koje predstavlja proveru podudarnosti obrazaca. Ove karakteristike čine Scalu idealnom za razvijanje aplikacija kao sto su web servisi, dok je kompatibilnost sa Javom cini pogodnom za razvoj android aplikacija.[7] Konkurentno programiranje zasnovano na akterima koji razmenjuju poruke, slično kao u jeziku Erlang.

3 Razvojno stablo

Na nastanak jezika Scala najviše je uticao programski jezik **Java**. Pored njega značajnu ulogu imali su programski jezici **OCaml** i **Erlang**, kao i mnogi drugi. Razvojno stablo jezika Scala može se videti na slici 2.



Slika 2: Razvojno stablo jezika Scala

3.1 Java

Objektno-orijentisani programski jezik Java nastaje 1995. godine pod okriljem kompanije **Sun Microsystems**. Glavni tvorac je **James Gosling**. Od svog nastanka pa do danas Java je jedan od najpopularnijih programskih jezika. Jedna od bitnijih karakteristika, koja je i dovela do nastanka ovog programskog jezika je platformska nezavisnost. Ova osobina omogućava lako prenošenje Java programa sa jedne platform na drugu, što je i bio razlog nastanka novog objektno-orijentisanog jezika. Jedan od problema u radu sa tada popularnim objektno-orijentisanim programskim jezikom `C++` je njegova platformska zavisnost. Java kod se prvo kompajlira do nivoa bajt-koda, koji je platformski neutralan, a potom i interpretira.To dovodi do postojanja jedinstvenog izvornog Java koda koji će se isto izvršavati na svim platformama. Glavne karakteristike ovog jezika su: objektna orijentisanost, jednostavnost, efikasnost, interpretacija, robustnost i sigurnost , višenitnost i dinamičnost, platformska nezavisnost koja povlači prenosivost i distribuiranost.[1] Zahvaljujući njima programski jezik Java ima široku primenu.
Kompatibilnost programskog jezika Scala sa Javom omogućava korišćenje svih pogodnosti koje Java pruža, dok podržanost funkcionalnog programiranja donosi spektar novih prednosti: eliminisanje bočnih efekata, kraci zapis, ... Svi Java tipovi imaju ekvivalent u Scala jeziku. Možemo koristiti sve Java biblioteke. Kompatibilnost omogućava lak prelazak iz Java u Scala kod,ali obrnuti prelazak nosi sa sobom potencijalne problem jer za neke delove Scala koda ne moramo imati ekvivalent u Javi (jedan od primera su trejtovi). [9]
Kao i u Javi i ovde se susrećemo sa klasama i njihovom hijerarhijom.Na vrhu klasne hijerarhije se nalazi klasa `scala.Any` koja ima dve potklase `scala.AnyVal` i `scala.AnyRef` koje predstavljaju nadklase klasa za vrednosti i klasa za reference. Može se reći da klasa `scala.AnyRef` odgovara klasi `java.lang.Object` u Javi, sve korisnički definisane klase ovde nasleđuju klasu `scala.AnyRef`. U Scali nemamo statičke promenljive i metode i zato postoje **singleton objekti** u koje stavljamo promenljive i metode koje bismo u Javi proglasili statičkim.[7] Oni mogu biti samostalni objekti ili povezani sa istoimenom klasom. Svaki program ima bar jedan singleton objekat sa `main` metodom. Klase mogu imati parametre. Za razliku od Jave, u Scali su sve vrednosti objekti. Na slikama 3 i 4 možemo uporediti ova dva jezika. Uvođenjem lambda izraza Java prisvaja određene koncepte funkcionalnog programiranja, ali i dalje nije funkcionalni jezik, za razliku od Scale.

```java
public class HelloWorld{
  public static void main(String[] args)
    {
    System.out.println("Hello World!");
  }
}
```

Listing 1: Java

```scala
object HelloWorld{
  def main(args: Array[String] ){
    println(" Hello World! ");
  }
}
```

Listing 2: Scala

Slika 3: 'Hello World' program u jezicima Java i Scala

```java
public class Student{
  public final String ime;
  public final int ocena;

  Person(String ime,int ocena){
    this.ime=ime;
    this.ocena=ocena;
  }
}
```

Listing 3: Java

```scala
class Student(val ime: String,
              val ocena: int)
```

Listing 4: Scala

Slika 4: Definisanje klase u jezicima Java i Scala

Logo jezika Java prikazan je na slici 5.

Slika 5: Logo programskog jezika Java

3.2 OCaml

OCaml je funkcionalan, imperativan, objektno-orijentisan programski jezik. Nastaje kao **Objective Camil** 1996.godine a potom je preimenovan 2011. godine. Svi delovi koda su upakovani u module. OCaml poseduje funktore koje možemo posmatrati kao funkcije iz modula u module. Kao što je funkcija parametrizovana vrednostima- argumentima, tako je i funktor parametrizovan modulima.[5]
Kao kombinacija funkcionalne i objektno-orijentisane paradigme OCaml je imao uticaj na porgramski jezik Scala. Objective Camil kombinuje ideje objektno orijentisanog i funkcionalnog programiranja sa ML-statičkim tipiziranjem. ML jezik je koren svih jezika koji pripadaju familiji strogo tipiziranih funkcionalnih jezika.[8] Logo jezika OCaml prikazan je na slici 6.

Slika 6: Logo programskog jezika OCaml

3.3 Erlang

Erlang je konkurentan, funkcionalan i deklarativan programski jezik, dinamički tipiziran i kompajliran. Nastaje 1986. pod okriljem firme **Erikson**. (engl. *Ericsson*) i dobija ime po danskom matematičaru Agneru Krarupu Erlangu (engl. *Agner Kraurp Erlang*). Preuzima sintaksu i mnoge koncepte iz programskog jezika Prolog, ali ih i obogaćuje konceptima konkurentne paradigme. Scala, kao i Erlang ima podršku za funkcionalno programiranje. Za nas je posebno interesantna konkurentnost programskog jezika Erlang čiji uticaj mozemo primetiti kod Scala jezika. Konkurentnost slanjem poruka, koja se ovde javlja, podrazumeva postojanje velikog broja procesa koji nikad ne dele memoriju, već komuniciraju isključivo asinhronim slanjem poruka.[2] Dok se konkurentno programiranje u programskom jeziku Scala zasniva na akterima koji razmenjuju poruke. Logo jezika Erlang prikazan je na slici 7.

Slika 7: Logo programskog jezika Erlang

3.4 .NET

.NET je razvojna platforma, obuhvata skup različitih tehnologija.
. NET Framework se sastoji od :

- Pet zvaničnih jezika: C#, VB, Visual C++, Visual J# i JScript.NET
- CLR-softversko okruzenje za izvrsavanje programa za .NET
- Brojne srodne biblioteke klasa-biblioteka klasa Frameworka (FCL) [3]

Juna 2004.Scala je modifikovana za .NET Framework ali od 2012.godine podrska za .NET je napuštena.

4 Zaključak

U ovom tekstu ukratko su predstavljene osnove razvoja programskog jezika Scala. Prikazani su jezici koji su najviše uticali na njegov nastanak i razvoj, i prikazano je njegovo elementarno razvojno stablo. Za dodatne informacije i praćenje daljeg razvoja, kao početnu odrednicu najbolje je koristiti zvaničnu stranu programskog jezika Scala[7].

Literatura

[1] Mirjana Ivanović, Zoran Budimac, Miloš Radovanović, and Dejan Mitrović. *Objektno-orijentisano programiranje i programski jezik Java,*. Siga star, 2016.

[2] Tijana Jevtić, Jelena Mrdak, David Dimić, and Zorana Gajić. Erlang-funkcionalno rešenje za konkurentni svet , 2019. Seminarski rad u okviru kursa Metodologija stručnog i naučnog rada.

[3] Jesse Liberty. *Programiranje na jeziku C#,*. Mikro knjiga, 2007.

[4] Ana D. Mitrović. Primene jezika Skala u paralelizaciji rasplinutog testiranja, 2019. Master rad.

[5] OCaml. Zvanična stranica programskog jezika OCaml. https://ocaml.org/.

[6] Martin Odersky, Lex Spoon, and Bill Venners. *Programming in Scala*. Artima Press, 2008.

[7] Scala. Zvanična stranica programskog jezika Scala. https://docs.scala-lang.org/.

[8] Tijana Todorov, Tamara Garibović, David Nedeljković, and Mihajlo Vićentijević. F# na .NET platformi , 2019. Seminarski rad u okviru kursa Metodologija stručnog i naučnog rada.

[9] Katedra za računarsku tehniku i informatiku. Sajt predmeta Funkcionalno programiranje na Elektrotehničkom fakultetu u Beogradu. http://rti.etf.bg.ac.rs/rti/ms1fp/.