Skript programiranje

Miloš Kutlešić , Miloš Đurić
mr16182@alas.matf.bg.ac.rs, mr16146@alas.matf.bg.ac.rs

4. Novembar 2018.

1 Uvod

U ogromnom svetu programskih jezika skript jezici su doživeli veliki "bum" jer im je jedna od glavnih namena ispomoć tradicionalnim programskim jezicima. Samim tim već vidimo da se skript jezici klasifikuju kao posebna grupa jezika, odvojena od tradicionalnih programskih.

Pre svega treba znati prednosti i mane skript jezika u odnosu na tradicionalne programske jezike i prepoznavati situacije u kojima ih je efikasnije i praktičnije koristiti. Za to je naravno potrebno iskustvo u programiranju, ali pre svega poznavanje raznolikosti primena programskih jezika, kao i njihovih karaktetistike.

2 Razlika između *skript* i tradicionalnih programskih jezika

Jedna od najvažnijih razlika između skript jezika i tradicionalnih programskih jezika je u tome što se tradicionalni programski jezici prvo tipično prevode (eng. compile) i zatim pokreću, dok su skript jezici interpretirani iz izvornog koda (eng. source code) ili bajt koda (eng. byte code) komandu po komandu (eng. runtime).[1]

To nam govori da skript jezike pretežno koristimo kada nam je brzina izrade bitnija od brzine izvršavanja, jer bi npr. potencijalno ponovno prevođenje većeg projekta uzelo dosta vremena u slučaju tradicionalnih programskih jezika.

Moglo bi se čak i postaviti drugačije pitanje: „Kada ne koristimo skript jezike?" i odgovor bi bio: „Kada su nam performanse bitnije". Takođe, glavni delovi aplikacija zahtevaju brzo i često izvršavanje pa se oni shodno tome pišu u tradicionalnim programskim jezicima, dok se oni delovi, koji se često menjaju i kojima bi potencijalno i korisnik pristupao i vršio izmene, pišu u skript jezicima. Kada bi uopštenije govorili, skript jezike bi koristili za automatizovanje određenih delova programa i izvlačenje informacija iz skupa podataka.

3 Tipovi skript jezika

Skript jezike delimo na :

- *Lepljivi*(eng. *Glue*) jezici koji kao što im i samo ime kaže služe povezivanju, tj. "lepljenju", softverskih komponenti (aplikacija, servera i slično).
- *Jezici za kontrolu poslova i ljuske* i Npr. služe startovanju sistemskog programa ili kontroli njegovog ponašanja.
- *GUI (eng.* Graphical User Interface*)* skript jezici predstavljaju interakciju sa grafičkim korisničkim interfejsom. Takođe se zovu i "makroi" zato što se većinski koriste da bi automatizovali neke radnje koje korisnik često ponavlja, tj. pišu se skripte koje se automatizovano ponavljaju.
- *Specifični jezici za aplikacije* se mogu podeliti na one koji su isključivo za jednu aplikaciju i na one koji su namenjeni radu u više sličnih aplikacija. Služe se opisivanje specifičnih akcija u aplikaciji i na taj način olakšavaju sam posao izrade aplikacije.
- *Ugradivi jezici* su slični jezicima za aplikacije iz prostog razloga što im je svrha da ih zamene u određenoj aplikaciji, tj. ostavlja se prostor pri programiranju aplikacije u tradicionalnom programskom jeziku gde skript jezik preuzima kontrolu nad aplikacijom. Može se reći i da su ovi jezici tehnički ekvivalentni produžecima specifičnih jezika za pojedine aplikacije, međutim većina programera teži već poznatim ovakvim jezicima kako bi već stečeno znanje mogli iskoristiti u što više aplikacija.
- *Skript jezici koji se koriste za izradu web stranica* imaju veliku primenu na internetu.
- *Komandni jezici operativnih sistema* , npr. bash i shell skripte.

4 Upotreba i istaknuti predstavnici

Jedna od bitnijih primena skript jezika jeste poboljšanje funkcionalnosti aplikacija, tj. olakšavanje implementacije rešenja. Za primer možemo uzeti korišćenje stringova pri manipulaciji sa regularnim izrazima ili obezbeđivanje lakšeg pristupa nižim nivoima operativnog sistema.

Danas se većinom koriste za izradu dinamičkih web stranica, da bi objasnili razliku i sličnosti između tradicionalnih programskih i skript jezika možemo uzeti primer jezika Java i JavaScript: [3]

JAVA

- *objektno orijentisan programski jezik*
- *aplikacije se pokreću pomoću JVM (eng. Java Virtual Machine) ili pretraživača*
- *klase su nužne*
- *prevodi izvorni u bajt kod koji se pokreće pomoću JVM*
- *samostalan jezik*
- *koristi mnogo više memorije*
- *koristi višenitni pristup množini istovremenih problema*

JAVA SKRIPT

- *objektno orijentisan skript jezik*
- *aplikacije se pokreću isključivo pomoću pretraživača*
- *koristi prototipsko nasleđivanje*
- *ne prevodi se, već se pokreće pomoću JavaScript interpretatora ugrađenog u pretraživaču*
- *deo web stranice i uklapa se sa njenim HTML sadržajem*
- *baš zbog toga što koristi manje memorije koristi se prilikom izrade web stranica*
- *sve se rešava pomoću jedne niti – event based pristup*

5 Prednosti i primeri

Jedna od prednosti skript jezika je to što nemaju tipove, tj. nije potrebno deklarisati tip promenljive jer će interpretator "zaključiti" kojeg je tipa promenljiva na osnovu vrednosti koja joj bude dodeljena. Predodređeni tip podataka je string. Međutim šta se dešava kada skript jezik treba pristupiti npr. strukturama i klasama jezika C ili C++? Problemi nastaju jer je skript jeziku teško da "razume" baš sve koncepte koje nam pruža tradicionalni programski jezik, u ovom slučaju C/C++, tj. interpreter ne zna šta da radi sa njima.

Primer 5.1 *Koncept nasleđivanja koji u skript jezicima ne postoji.[2]*

Takve probleme možemo rešiti na sledeći način:

```
struct Vector {
Vector();
~Vector();
double x,y,z;
};
```

Kod iznad možemo transformisati u sledeći:

```
Vector *new_Vector();
void delete_Vector(Vector *v);
double Vector_x_get(Vector *v);
double Vector_y_get(Vector *v);
double Vector_z_get(Vector *v);
void Vector_x_set(Vector *v, double x);
void Vector_y_set(Vector *v, double y);
void Vector_z_set(Vector *v, double z);
```

Stoga interpretator prethodne funkcije može gledati kao:

```
% set v [new_Vector]
% Vector_x_set $v 3.5
% Vector_y_get $v
% delete_Vector $v
% ...
```

S obzirom da je u pristupnim funkcijama integrisan mehanizam pristupa objektu i njegovim poljima, za interpretator nije bitno kako se vektor zapravo predstavlja.

Proksi klase (eng. proxy classes) su posebna vrsta klasa koje se u skript jeziku prave da bi obezbedile pristup gore spomenutim klasama ili strukturama, tj. kako im ime kaže one ih zastupaju. Ako definišemo strukturu vektora na sledeći način u C-u:

```
class Vector {
public:
Vector();
~Vector();
double x,y,z;
};
```

Tada ćemo u Pajtonu (eng. Python) zbog proksi mehanizma moći mnogo prirodnije raditi sa interpretatorom:

```
>>> v = Vector()
>>> v.x = 3
>>> v.y = 4
>>> v.z = -13
>>> ...
>>> del v
```

Prilikom korišćenja proksi klasa koriste se zapravo 2 objekta, jedan koji koristi skript jezik i koji je zapravo „radni“, i onaj drugi koji „vučemo“ iz C/C++.

6 Korišćenje memorije

U tradicionalnim programskim jezicima se od nas većinom zahteva da ručno alociramo memoriju za promenljive, dok je u skript jezicima implementiran sakupljač otpadaka (eng. garbage collector), uzećemo primer u JavaScript jeziku.

Razlog implementiranja nečeg poput sakupljača otpadaka je baš taj što curenje memorije (eng. memory leak) i većina drugih problema vezana za memorije upravo nastaju pri neefikasnom oslobađanju iskorišćene memomirje. Sakupljač otpadaka je zapravo proces koji pronalazi delove memorije, zauzete naravno, i oslobađa ih. Za to može koristiti neke od sledećih algoritama:

- Reference - counting garbage collection
- Mark-and-sweep algoritam

„Reference – counting garbage collection“ je najbitniji algoritam koji koristi sakupljač otpadaka koji sakuplja sve one objekte na koje više nema referenci i „gleda“ ih kao one koje aplikacija više ne koristi. Drugi spomenuti algoritam ide malo drugačijom logikom, naime on prvo pronalazi sve globalne objekte i redom ih „vezuje“, tj. traži reference na početni objekat, zatim reference na te reference, itd. Zatim ih deli na one do kojih može i ne može doći (eng. reachable/unreachable objects). Oni do kojih nije uspeo doći će automatski biti „sakupljeni“, tj. memorija koju zauzimaju će biti oslobođena. Ovaj algoritam je superiorniji od prethodnog jer će on ujedno „sakupiti“ i one objekte na koje nema više referenci, tj. „pokriće“ i prethodni algoritam.

7 Zaključak

Kao što je i navedeno, skript programski jezici doživljavaju sve veću ekspanziju zbog jednostavnijeg korišćenja (jednostavna semantika i sintaksa, manje koda), i do 4 puta manje vremena se potroši u proseku prilikom razvoja aplikacija. Samim tim se i povećava produktivnost programera, a brži razvoj aplikacija doprinosi i bržem izvršavanju istih.

Literatura

[1] Scripting language,https://www.techopedia.com/definition/3873/scripting-language

[2] Web Development,https://levelup.gitconnected.com/web-development-languages-36241b046a81

[3] Difference between java and javascript,https://www.geeksforgeeks.org/difference-between-java-and-javascript/