Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/26BE32Sorgic

PODRŠKA ZA PYFLIES JEZIK U VS CODE EDITORU

SUPPORT FOR PYFLIES LANGUAGE IN THE VS CODE EDITOR

Dejan Šorgić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu implementirana je nadogradnja na postojeću ekstenziju za Visaul Studio Code okruženje koja pomaže pri pisanju eksperimenata u PyFlies jeziku. U okviru rada je primarno implementiran jezički server, koji pomoću Language Server Protocol-a komunicira sa PyFlies ekstenzijom i tako proširuje njene funkcionalnosti.*

Ključne reči: *ekstenzija, pyFlies, DSL, Language Server Protocol, Visual Studio Code, jezički server*

Abstract – *This paper details the implementation of an enhancement to the pre-existing Visual Studio Code extension designed to facilitate experiment scripting in the PyFlies language. The central focus of this study is a language server, which uses Language Server Protocol to communicate with PyFlies extension and with this, it extends its functionalities.*

Keywords: *Extension, pyFlies, DSL, Language Server Protocol, Visual Studio Code, language server*

1. UVOD

Ovaj rad obuhvata dizajn i implementaciju nadogradnje na podršku za *Visual Stuio* Code radno okruženje za *PyFlies* jezik [1]. Inicijalna verzija *PyFlies* ekstenzije obuhvata bazične funkcionalnosti, a u okviru ovog rada je implementiran niz kompleksnih funkcionalnosti, kao što su analiza koda i prijava grešaka, skok na definiciju, prikaz referenci i još neke. Za implementaciju pomenutih funkcionalnosti, potrebno je osloniti se na *Language Server Protocol*, i napraviti jezički server. Jezički server i inicijalna verzija ekstenzije, koja u ovom kontekstu predstavlja klijenta, čine dve komponente ekstenzije, koje komuniciraju preko *LSP*-a.

PyFlies je jezik specifičan za domen, i njegova namena je pisanje kognitivnih testova. Zamišljen je tako da bude lako čitljiv i jednostavan za učenje. Jezik je baziran na *Python* programskom jeziku i sadrži modularnu arhitekturu, kako bi generatori mogli biti razvijeni kao odvojeni projekti. Trenutno dostupni generatori su *csv*, *log* i *psychopy* generator, od kojih su *csv* i *log* generatori ugrađeni u sam *PyFlies* projekat, dok je *psychopy* generator u odvojenom projektu.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Igor Dejanović, red. prof.

PsychoPy [2] je platforma za pisanje i izvršavanje kognitivnih eksperimenata, a *psychopy* generator omogućava generisanje *Python* skripti koje se ovde mogu koristiti. Generisanje *log* i *csv* datoteka na osnovu *PyFlies* eksperimenta se izvršava u cilju analize, pregleda toka eksperimenta i pronalaska grešaka u eksperimentu, u slučaju *log* datoteka, odnosno u cilju ekstrakcije tabele uslova definisane u eksperimentu, u slučaju *csv* datoteka.

Dizajn *Pyflies* jezika je vođen sledećim principima: čitljivost, apstrakcija, koncepti domena, modularna arhitektura i proširivost, najbolje prakse i fokus na strukturu eksprimenta.

Praćenju ovih principa idu u korist činjenice da je *PyFlies* jednosavan za razumeti i lak za naučiti, i to da se za opis eksprimenata koriste apstraktni pojmovi, i usko povezani za domen, bez brige o detaljima implementacije, što umanjuje mentalni napor potreban za pisanje i razumevanje eksperimenta [1].

2. JSD I *VS CODE* PODRŠKE

Jezici specifični za domen, JSD (en. *Domain Specific Languages, DSL*) predstavljaju programske jezike ograničene ekspresivnosti, dizajnirane za rešavanje specifičnih problema u određenoj oblasti [1]. Ovi jezici konstruišu se tako da odgovaraju specifičnim potrebama domena, pružajužajući prilagođene koncepte, sintaksu i semantiku za izražavanje ideja i operacija u tom domenu. JSD-ovi menjaju generalizaciju za ekspresivnost u ograničenom domenu [4].

Prednosti korišćenja JSD-ova uključuju povećanu produktivnost u određenom domenu, smanjen napor za izražavanje ideja i implementaciju rešenja, kao i smanjeni broj linija programskog koda potreban za rešenje. Takođe, znatno je poboljšana čitljivost i umanjena mogućnost grešaka.

Visual Studio Code, često skraćeno sa ***VS Code***, je besplatan tekstualni editor otvorenog koda (en. *open source*). Razvijen je od strane *Microsoft*-a, i namenjen je za operativne sisteme *Windows*, *Linux* i *macOS*. Ima ugrađenu podršku za programske jezike *JavaScript* i *TypeScript* i dalje omogućava podršku za druge jezike kroz bogat ekosistem ekstenzija. Kako *VS Code* koristi sistem foldera i radnog prostora (en. *workspace*) da interaguje sa projektom i sa sistemom dokumenata za manipulisanje tekstualnim fajlovima koji sadrže kod, ovo je omogućilo da se funkcionalnosti specifične za jezik dodele na osnovu ekstenzije fajla.

2.1. Jezičke ekstenzije u *VS Code*

VS Code ne obezbeđuje ugrađenu podršku za jezike, ali nudi set *API*-ja koji omogućavaju bogate funkcionalnosti za jezike. Tržište ekstenzija za *VS Code* (en. *VS Code marketplace*), predstavlja centralizovanu platformu koja omogućava programerima da istraže, preuzmi i integrišu različite ekstenzije u svoje razvojno okruženje.

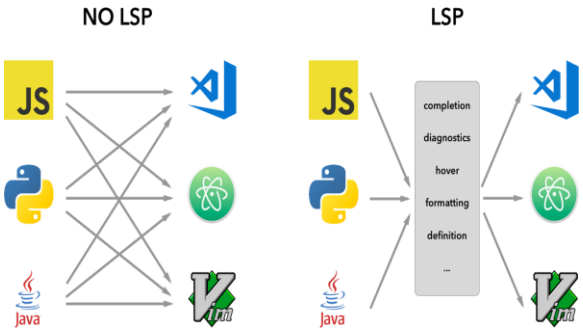
Postoje okvirno dve kategorije jezičkih funkcionalnosti;

Deklarativne jezičke funkcionalnosti – pretežno se bave definisanjem i određivanjem ponašanja ekstenzije putem struktuiranih konfiguracija. Dopušta programerima da izraze šta bi ekstenzija trebala raditi, bez potrebe da se detaljno upuštaju u implementaciju. Neki od primera ovih funkcionalnosti su: bojenje sintakse (en. *syntax highlighting*); automatsko dovršavanje koda (en. *snippet completion*); uparenost zagrada (en. *bracket matching*); automatsko zatvaranje zagrada (en. *bracket autoclosing*) itd.

Programske jezičke funkcionalnosti – omogućavaju programerima da pišu kod koji direktno interaguje sa *VS Code* editorom i njegovim osnovnim *API*-ima. Ovakve funkcionalnosti često su podpomognute Jezičkim serverom, programom koji analizira kod napisan u projektu i dinamički izvršava korisne akcije. Neki od primera programskih jezičkih funkcionalnosti su: prikaz informacija pri prelasku mišem (en. *hover information*); automatsko završavanje koda (en. *auto completion*); skok na definiciju (en. *jump to definition*); provera grešaka (en. *error checking*) itd.

3. ARHITEKTURA PYFLIES PODRŠKE

Jezičke podrške koje se oslanjaju na Protokol jezičkog servera (en. *Language server protocol*, *LSP*) predstavlja ključan napredak u oblasti integrisanih razvojnih okruženja i uređivača koda. To je otvoren i standardizovan protokol koji olakšava interakciju između uređivača koda i jezičkih servera, odvojenih procesa koji se bave analizom programskih jezika. *LSP* je, pre svega, uveden kako bi se rešio izazov pružanja bogate podrške za programske jezike u različitim uređivačima koda i *IDE*, budući da je održavanje jezičkih funkcionalnosti za svako pojedinačno razvojno okruženje postalo nemoguće.



Slika 1. Razlika u nekorišćenju i korišćenju LSP-a [5]

Na slici 1 se može videti ilustracija ovoga, da uvođenjem međusloja, u vidu Protokola jezičkog servera, postaje dovoljno da se podrška za određeni jezik implementira samo jednom, umesto za svaki *IDE* posebno.

Komunikacija u okviru *LSP*-a se odvija pomoću *JSON-RPC* i ovaj protokol definiše skup poruka koje omogućavaju efikasnu komunikaciju između jezičkog servera i klijena. Ove poruke se svrstavaju u tri glavna tipa: poruke zahteva (en. *request messages*), obaveštenja (en. *notification messages*) i poruke odgovora (en. *response messages*).

4. OPIS FUNKCIONALNOSTI EKSTENZIJE

Prvi korak ka korišćenju *PyFlies* podrške u *VS Code*-u je instalacija odgovarajuće ekstenzije. Ekstenzija je dostupna na *VS Code* tržištu, a korisnik može pretraživati i instalirati direktno iz radnog okruženja. Kada korisnik otvori dokument napisan u *PyFlies* jeziku, što će ekstenzija prepoznati kao bilo koji fajl sa nastavkom ".pf", ekstenzija će se automatski aktivirati i odmah će izvršiti određene akcije, uključujući bojenje sintakse i analizu koda.

Bojenje sintakse je jedna od primarnih funkcionalnosti *PyFlies* podrške, i to je prva stvar koju korisnik primeti pri otvaranju fajla napisanog u *PyFlies* jeziku. Ova funkcionalnost ima za cilj da postavi boju i stil na izvorni kod kako bi se olakšalo čitanje i razumevanje.

Analiza koda je takođe nešto što se izvršava pri otvaranju *PyFlies* fajla, i ukoliko ima grešaka, to će biti prijavljeno korisniku. Problematični delovi koda će biti podvučeni crvenom linijom, a pri pomeranju kursora miša preko označenog dela koda, korisniku će biti prikazan razlog za grešku.

Na slici 2 se može videti primer prijave greške u kodu, kao i funkcionalnost bojenja sintakse.

```
65
66      Press SPACE for the real block.
67  }
68
69  flow {
70    show Practicez
71    execute Simon(practice true)
72
73    show Real
74    execute Simon(repetitions 10, random true)
75
76  }
```

Unknown object "Practicez" of class "ScreenType" pyFlies LS
View Problem (Alt+F8) No quick fixes available
You, 5 seconds ago • Uncommitted changes

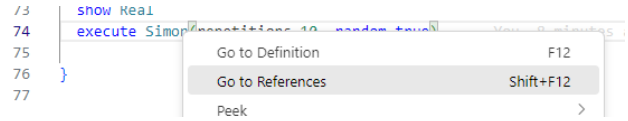
Slika 2. Primer prijave greške u izvornom kodu

Jedna od funkcionalnosti koja značajno olakšava proces pisanja koda je automatsko kompletiranje koda, što dovodi do smanjenja broja unetih karaktera, kao i grešaka pri kucanju. *PyFlies* podrška će prepoznati da li postoje delovi koda (en. *snippets*), koji mogu biti ubačeni na mestu gde se nalazi kursor, u zavisnosti od konteksta i unetok karaktera. Ako postoje predlozi, korisniku će biti prikazana lista opcija ispod koda koji se unosi. Korisnik može navigirati kroz listu predloga i izabrati neki od preloga. Izborom jedne od opcija, ceo isečak koda će biti ubačen u izvorni kod, s tim što će se od korisnika očekivati da unese smislene vrednosti promenljivih, ukoliko izabrani isečak ima jednu ili više promenljivih u sebi.

Kada je reč o definisanim promenljivima, potrebno je pomenuti funkcionalnosti „skok na definiciju“ i „prikaz referenci“. Skok na definiciju je funkcionalnost koja omogućava korisniku da se sa bilo koje lokacije u dokumentu prebaci na lokaciju gde počinje definicija određene promenljive ili rutine. Sa druge strane, funkcionalnost prikaza referenci omogućava korisniku da vidi sve lokacije u izvornom kodu na kojima se koristi konkretna promenljiva ili rutina. Rezultat ove akcije će

biti specifičan prikaz, koji se u *VS Code*-u naziva prozor virenja (en. *peek window*), čiji je cilj da omogući korisniku da ostane u istom dokumentu, i na istoj lokaciji u dokumentu, pružajući uvid u sve druge lokacije gde se koristi tražena promenljiva ili rutina.

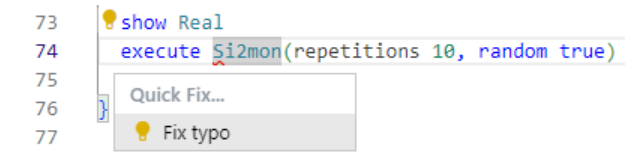
Na slici 3 se mogu videti opcije koje omogućavaju izvršavanje pomenutih funkcionalnosti.



Slika 3. Prikaz načina da se izvrše akcije skok na definiciju i prikaz referenci

Poslednja funkcionalnost koju je važno pomenuti je brza ispravka (en. *quick fix*). Ukoliko postoji greška u izvornom kodu i na pošetku linije koda na kojoj je prijavljenja greška se pojavi ikonica sijalice, to je znak da je dostupna brza ispravka.

Klikom na ikonicu sijalice, dobija se prikaz opcija, kao što se može videti na slici 4. Trenutna verzija podrške za *PyFlies* nudi jednu vrstu brze ispravke, a to je ispravka greške u kucanju kada korisnik unosi referencu na neku promenljivu ili rutinu.



Slika 4. Prikaz ponude brze ispravke za grešku u kucanju

5. IMPLEMENTACIJA

U okviru ovog rada, serverska strana ekstenzije je kompletno implementirana, dok se na klijentsku stranu nadogradilo. Za implementaciju jezičkog servera korišćena je *pygls* biblioteka, kao i *lsprotocol*, paket koji predstavlja *Microsoft*-ovu implementaciju tipova objekata koji se koriste u okviru *LSP*-a.

Najbitniji dodatak klijentskoj strani ekstenzije je vezan za konfiguraciju jezičkog servera, instaliranje servera pri prvom pokretanju ekstenzije i pokretanje servera svaki sledeći put kada se aktivira ekstenzija.

Jezički server za *PyFlies* podršku je kreiran kao odvojen projekat u koji je smeštena cela logika. Taj projekat je postavljen na *PyPi* repozitorijum, i sa tog repozitorijuma je moguće instalirati *PyFlies* jezički server pokretanjem komande:

```
pip install pyflies-ls
```

Trenutna verzija *pyflies-ls* paketa, postavljena na *PyPi*, je 0.0.3, i postavljena je da zahteva pygls paket, verzije 1.0.2, i pakete *textx* i *pyflies*, bez konkretno određene verzije paketa.

Analiza napisanog programskog koda i prijava grešaka, kako sintaksnih, tako i semantičkih, predstavlja jednu od najvažnijih funkcionalnosti ove ekstenzije. Takođe, predstavlja jednu od kompleksnijih sa strane implementacije.

```
@pyflies_server.feature(TEXT_DOCUMENT_DID_CHANGE)
def did_change(ls, params: DidChangeTextDocumentParams):
    """
    Text document did change notification.
    The method calls validation on the document in which the change occured.
    """
    _validate(ls, params)


@pyflies_server.feature(TEXT_DOCUMENT_DID_OPEN)
async def did_open(ls, params: DidOpenTextDocumentParams):
    """
    Text document did open notification.
    The method calls validation on the opened document.
    """
    _validate(ls, params)
```

Listing 1. Registracija događaja otvaranja i izmene dokumenta

Kao što se može videti na listingu 1, implementira se registracija događaja otvaranja i izmene dokumenta, i kao odgovor na te događaje, poziva se *_validate* metoda.

Metoda *_validate* prvo instancira izvorni kod dokumenta, koristeći *load_document_source* metodu, koja se oslanja na mogućnosti koje nudi *pygls* biblioteka, da učita izvorni kod. Zarim se poziva metoda validate, gde se nalazi čitava logika vezana za analizu. Analiza se vrši pozivom metoda importovanih iz *textX*-a. Pozivom metode *metamodel_for_language* i proleđivanjem parametra PyFlies instancira se metamodel za jezik PyFlies. Za taj metamodel se poziva metoda *model_from_str*, kojoj se proleđuje izvorni kod. Cilj pozivanja ove metode jeste taj da izazovemo *PyFliesException* ili *TextXError* u slučaju da izvorni kod ima neke nepravilnosti. Ukoliko nepravilnosti postoje, kreiraće se objekat *Diagnostic* klase, koji u sebi sadrži *Range* objekat, koji će pokazati tačnu liniju u izvornom kodu koja uzrokuje nepravilnost, poruku sa opisom toga šta je uzrok nepravilnosti, i source vrednost, koja je uvek postavljena na *PyFlies LS*, kako bi se znalo koji je izvor dijagnostike. U suprotnom, kada je kod ispravan, metoda validate kao rezultat vraća praznu listu grešaka.

```
def validate(model: str) -> List[PyFliesException]:
    errors = []
    try:
        mm = metamodel_for_language("pyflies")
        mm.model_from_str(model)
    except PyFliesException as err:
        # TODO: How to determine col and line for PyFliesException
        errors.append(construct_diagnostic(err.args[0], 1, 1))
    except TextXError as err:
        msg = err.message
        col = err.col
        line = err.line


        errors.append(construct_diagnostic(msg, col, line))
    except Exception as e:
        print(e)
    return errors
```

Listing 2. Prikaz dela koda koji vrši validaciju izvornogPyFlies koda

Na listingu 2 može se videti opisani deo koda. Dakle, *PyFlies LS* će na događaj otvaranja ili promene u fajlu odgovoriti sa listom, koja će sadržati *Diagnostic* objekat ili će biti prazna. *LSP* će znati kako da interpretira

Diagnostic objekat na klijentskoj strani ekstenzije, pošto je to objekat *lsprotocol* paketa.

Automatsko kompletiranje je implementirano dodavanjem metode completions, koja kao parametre prima instancu jezičkog servera i *CompletionParams* objekat. Proces izbora isečaka koda koji će biti ponuđeni se izvršava učitavanjem dokumenta, uzimanjem karaktera koji je korisnik uneo i filtriranjem mogućih isečaka tako da ostanu oni koji odgovaraju unetom karakteru. Sledeći korak je prosleđivanje učitanog dokumenta i filtriranih isečaka koda metodi koja će da instancira *PyFlies* model. Iteracijom kroz kandidatske iseče koda, radi se provera za svaki, tako što se na odgovarajuću poziciju u dokumentu ubaci isečak i proverava se da li će instancirani model izazvati grešku, koja će biti tipa *TextXError*, ili ne. Ako ne dođe do izuzetka, znači da je isečak u tom delu koda, u tom kontekstu, validan i može vidi ubačen. Klijentu će se vratiti lista *CompletionItem* objekata, ili prazna lista, ukoliko nema validnih isečaka za automatsko kompletiranje.

Funkcionalnosti **skok na definiciju** i **pronalazak reference** implementirane su na sličan način. Prvi deo logike je vezan za uzimanje potpunog naziva simbola ili rutine za koju se izvršava ova operacija. To se radi tako što od pozicije kursora, koji se dobije kao podatak u zahtevu, ide ulevo i udesno, kako bi se prikupili svi karakteri koji čine naziv simbola. Kada je taj deo odrađen, u slučaju skoka na definiciju, traži se pozicija u dokumentu gde je definisana promenljiva ili rutina, pomoću instanciranja modela, što omogućava *textX*, sa prosleđenim izvornim kodom. Pristupanjem atributima instanciranog modela, može se doći do lokacije u dokumentu na kojoj je definsana promenljiva ili rutina. Taj podatak se klijentu vraća kao *Location* objekat.

Za prikaz referenci je razlika u tome što se koristi metoda *get_children*, takođe importovana iz *textX*-a, pomoću koje se formira lista *Location* objekata za traženu promenljivu ili rutinu.

Brza ispravka podrazumeva da postoji greška u kodu, tako da je prvi korak provera da li je to slučaj, i ukoliko nije, metoda završava, ne vraćajući odgovor. Ukoliko postoji greška u kodu, poziva se metoda koja će, po trenutnoj implementaciji, proveriti da li postoji greška koja ukazuje na nepostojeći, to jest nedefinisani objekat. Ovo se tretira kao greška u kucanju, u smislu da je namera bila da se unese referenca na neku od definisanih promenljivih ili rutina. Sa time na umu, iterira se kroz definisane nazive za odgovarajući tip objekta, i ukoliko postoji naziv definisanog objekta koji je dovoljno sličan onome što je korisnik uneo, a što je izazvalo grešku, onda se kreira *CompletionItem* objekat koji će zameniti uneti naziv promenljive sa ispravnim nazivom. Ukoliko ne postoji dovoljno dobar predlog za brzu ispravku, server neće vratiti ništa kao opciju.

6. ZAKLJUČAK

Tema ovog rada bila je opis podrške za *PyFlies* jezik u *VS Code* editoru, i u okviru istog pruženo je duboko razumevanje ključnih aspekata implementacije ove podrške. Opisan je *PyFlies* jezik, *VS Code* radno okruženje, i arhitektura podrške koja koristi Protokol jezičkog servera. Sa osloncem na opisane pojmove, predstavljen je niz funkcionalnosti koje *PyFlies* podrška nudi, i opisana implementacija tih funkcionalnosti.

Održavanje *VSCode* ekstenzije i jezičkog servera je u direktnoj zavisnosti od izmena na *PyFlies* jeziku. Ukoliko u budućnosti dođe do promena u gramatici jezika, u smislu da se dodaju nova pravila ili da se izmene postojeća, potrebno je prilagoditi *snippets\PyFlies-snippets.json* fajl u jezičkom serveru, kako bi automatsko kompletiranje bilo u skladu sa najnovijom verzijom *PyFlies* jezika.

Takođe, kod održavanja treba imati na umu verzije paketa koji su potrebni kako bi jezički server mogao da funkcioniše. Tu konkretno spadaju *PyFlies* paket, podrazumevano, kao i *textX* i *pygls*.

Nije retka pojava da se u novijim verzijama paketa neki modul ili metoda preimenuje, promeni ili potpuno obriše.

Problem sa verzijama paketa od kojih jezički server zavisi se može rešiti navođenjem tačnih verzija koje će biti iskorišćenje pri instalaciji. Međutim, novije verzije pomenutih paketa mogu doneti benefite u vidu novih funkcionalnosti ili boljih performansi, pa to treba razmotriti, kada je u pitanju dalji razvoj jezičkog servera.

7. LITERATURA

[1] Dejanović, I.; Dejanović, M.; Vidaković, J.; Nikolić, S. PyFlies: A Domain-Specific Language for Designing Experiments in Psychology. Appl. Sci. 2021, 11, 7823. https://doi.org/10.3390/app11177823

[2] Peirce, J., Gray, J.R., Simpson, S. et al. PsychoPy2: Experiments in behavior made easy. Behav Res 51, 195–203 (2019). https://doi.org/10.3758/s13428-018-01193-y

[3] http://textx.github.io/textX/stable/, pregledano 16. oktobar 2019.

[4] Mernik, M., Heering, J., & Sloane, A. M. (2005). When and how to develop domain-specific languages. ACM Computing Surveys, 37(4), 316–344. doi:10.1145/1118890.1118892

[5] Microsoft. 2018. Example - Language Server. (2018). https://code.visualstudio.com/api/language-extensions/language-server-extension-guide

Kratka biografija:

Dejan Šorgić rođen je u Beogradu 1996. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo i informacione tehnologije odbranio je 2023.god.
kontakt: dejans1224@gmail.com