Institut za matematiku i informatiku
Prirodno-matematicki fakultet
Univerzitet u Kragujevcu

Dart programski jezik

Student: Jelena Gogić(56/2018)

Sadrzaj

O Dart-u

Dart je objektno orijentisani programski jezik optimiziran za klijente na raznim platformama. Razvio ga je Google, a koristi se za izradu mobilnih, desktop, backend I web aplikacija. Skalabilna je I kohezivna platforma za izgradnju aplikacija. Upotrebite jezik Dart, biblioteke I alate da biste napravili bilo sta, od jednostavnih skripti do gotovih aplikacija s mnostvo opcija.

Dart je programski jezik koji se koristi za kodiranje Flutter aplikacija. Predstavlja softver otvorenog koda pod BSD licencom. Malo podseca na programski jezik C. Kada se koristi u web aplikacijama, prevodi se u JavaScript pa se moze pokretati na svim web pretrazivacima. Dart koristi "hot reload" kako biste odmah videli rezultat u vasoj pokrenutoj aplikaciji.

Dart-ovi ciljevi dizajna su:

- Stvoriti struktuirani, ali fleksibilni jezik za veb programiranje.
- Uciniti da se Dart programerima ucini poznatim I prirodnim, a time I lakim za ucenje.
- Uveriti se da Dart pruza visoke performance u svim modernim veb pretrazivacima I okruzenjima, od malih rucnih uredjaja do izvrsenja na strani servera.

Dart cilja sirok spektar razvojnih scenarija: od projekta za jednu osobu bez velike strukture do projekta velikog obima koji zahteva formalne tipove u kodu za navodjenje namera programera. Da bi podrzao ovaj sirok spektar projekata Dart ima opcione tipove: to znaci da mozete poceti sa kodiranjem bez tipova I dodati ih kasnije po potrebi.

Upotreba

Postoji cetiri glavna nacina za izvrsavanje Dart koda:

1. Preveden kao JavaScript

   Da bi se pokrenuo u obicnom veb pretrazivacu Dart se oslanja na izvor-u-izvor kompajliranje (source-to-source) u JavaScript. Prema sajtu projekta "Dart je dizajniran da bude lak za pisanje razvojnih alata, pogodan savremenom razvoju aplikacija I sposoban za implementacije visokih performansi. Kada je pokrenut u veb pretrazivacu Dart se prevodi u JavaScript pomocu dart2js kompajler. Kompajliran kao JavaScript Dart kod je kompatibilan sa svim glavnim pretrazivacima bez potrebe da pretrazivaci implementiraju Dart. Kroz optimizaciju prevedenog JavaScript izlaza, izbegavajuci skupe provere I operacije, kod napisan u Dartu moze u nekim slucajevima raditi brze nego ekvivalentni kod pisan koristeci JavaScript idiome.

2. U Dart-ijum pretrazivacu

   Dart Softver Development Kit(SDK) se isporucuje sa verzijom Kroumium veb pretrazivaca, modifikovanim da sadrzi Dart virtuelnu masinu (VM). Ovaj pretrazivac moze pokrenuti Dart kod direktno bez prevodjenja u JavaScript. Namenjen je kao razvojni alat za aplikacije napisane u tom jeziku, a ne kao veb pretrazivac opste namene. Prvo je planirano da se podrska za Dart ugradi direktno u Gugl Kroum, ali je kasnije obustavljeno.

3. Samostalno

   Dart SDK se isporucuje sa samostalnom Dart VM, omogucavajuci da se Dart kod pokrene u okruzenju interfejsa komandne linije. Kao jezik alati koji su ukljuceni u Dart SDK su napisani uglavnom u Dartu, samostalni Dart VM je kriticni deo SDK. Ovi alati ukljucuju dart2js kompajler I paket upravljanja pod nazivom pab. Dart se isporucuje sa kompletnom standardnom bibliotekom koja omogucava korisnicima da pisu potpuno funkcionalne sistemske aplikacije, kao sto su prilagodjeni veb server.

4. Unapred kompajliran

   Dart kod moze biti AOT-kompajliran u masinski kod. Aplikacije napravljene koristeci "Flutter", SDK aplikacijom za mobilne uredjaje izgradjenom na Dart-u, postavljenje su na prodavnice aplikacija kao AOT-kompajliran Dart kod.

Kodiranje u Dart-u

Dart je kao i Java, C, JavaScript, C# I drugi jezici, potomak ALGOL porodice jezika. Sintaksa kaskade metoda, koja pruza sintaksnu precicu za povezivanje nekoliko metoda jednog za drugim na isti objekat, usvojena je od strane Smalltalk-a. Dart koristi izolate kao vlasnicke I zastitne jedinice pri strukturisanju aplikacija. Koncept izolata nadogradjuje se na model Aktora, koji je najpoznatiji u jeziku Erlang.

Primeri:

1. Hello World

   Svaka aplikacija ima main() funkciju. Da biste prikazali tekst na konzoli mozete da koristite funkciju najviseg nivoa print().

```
void main() {
  print('Hello, World!');
}
```

2. Promenljive

Cak I u tipicnom Dart kodu, vecini promenljivama nisu potrebni eksplicitni tipovi, zahvaljujuci zakljucivanju tipa:

```
var name = 'Voyager I';
var year = 1977;
var antennaDiameter = 3.7;
var flybyObjects = ['Jupiter', 'Saturn', 'Uranus', 'Neptune'];
var image = {
  'tags': ['saturn'],
  'url': '//path/to/saturn.jpg'
};
```

3. Funkcije

Preporucuje se da navedete tipove argumenata svake funkcije i povratnu vrednost:

```
int fibonacci(int n) {
  if (n == 0 || n == 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}

var result = fibonacci(20);
```

4. Komentari

Dart komentari obicno pocinju sa //.

```
// This is a normal, one-line comment.

/// This is a documentation comment, used to document libraries,
/// classes, and their members. Tools like IDEs and dartdoc treat
/// doc comments specially.

/* Comments like these are also supported. */
```

5. Uvoz

Da biste pristupili API-ima definisanim u drugim bibliotekama, koristite import .

```
// Importing core libraries
import 'dart:math';

// Importing libraries from external packages
import 'package:test/test.dart';

// Importing files
import 'path/to/my_other_file.dart';
```

6. Klase

Evo primera klase sa tri svojstva, dva konstruktora I metodom. Jedno od svojstava ne moze se direktno podesiti, pa se definise pomocu metode getera (umesto promenljive).

```
class Spacecraft {
  String name;
  DateTime launchDate;

  // Constructor, with syntactic sugar for assignment to members.
  Spacecraft(this.name, this.launchDate) {
    // Initialization code goes here.
  }

  // Named constructor that forwards to the default one.
  Spacecraft.unlaunched(String name) : this(name, null);

  int get launchYear =>
      launchDate?.year; // read-only non-final property

  // Method.
  void describe() {
    print('Spacecraft: $name');
    if (launchDate != null) {
      int years =
          DateTime.now().difference(launchDate).inDays ~/
              365;
      print('Launched: $launchYear ($years years ago)');
    } else {
      print('Unlaunched');
    }
  }
}
```

7. Nasledjivanje

Dart ima jedno nasledstvo.

```
class Orbiter extends Spacecraft {
  double altitude;
  Orbiter(String name, DateTime launchDate, this.altitude)
      : super(name, launchDate);
}
```

8. Interfejsi I apstraktne klase

Dart nema kljucnu rec za interfejs. Umesto toga, sve klase implicitno definisu interfejs. Stoga mozete primeniti bilo koju klasu.

```
class MockSpaceship implements Spacecraft {
  // ···
}
```

Mozete napraviti apstraktnu klasu koju ce prosiriti (ili primeniti) konkretna klasa. Apstraktne klase mogu sadrzati apstraktne metode (sa praznim telima).

```
abstract class Describable {
  void describe();

  void describeWithEmphasis() {
    print('=========');
    describe();
    print('=========');
  }
}
```

9. Mixins

Mixins u nacin ponovne upotrebe koda u vise hijerarhija klasa. Sledi mesovita deklaracija:

```
mixin Piloted {
  int astronauts = 1;
  void describeCrew() {
    print('Number of astronauts: $astronauts');
  }
}
```

Da biste klasi dodali mogucnost kombinacije, samo prosirite klasu kombinacijom.

```
class PilotedCraft extends Spacecraft with Piloted {
  // ...
}
```

Foreign function interface

Interfejs strane funkcije (FFI) je mehanizam pomocu kojeg program napisan na jednom programskom jeziku moze pozivati rutine ili koristiti usluge napisane na drugom. Dart mobilne, komandne linije I serverske aplikacije pokrenute na Dart Native platformi mogu koristiti biblioteku dart:ffi za pozivanje C API-ja.

Sledeci primeri pokazuju kako se koristii biblioteka dart:ffi:

- hello_world - Kako pozvati funkciju C bez argumenata I bez povratne vrednosti.
- primitives - Kako pozvati C funkciju koje imaju argumente I vracaju vrednosti koje su celobrojne ili pokazivaci.
- structs - kako se koriste structure za prosledjivanje stringova u I iz C i za rukovanje slozenim I jednostavnim C strukturama.