Princípy tvorby softvéru
Programovacie paradigmy

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

Čo to je programovacia paradigma

A programming paradigm is a style, or "way," of programming.

Paradigm can also be termed as method to solve some problem or do some task. Programming paradigm is an approach to solve problem using some programming language or also we can say it is a method to solve a problem using tools and techniques that are available to us following some approach.

Programovacie paradigmy

- Je toho veľa, klasifikácia je nie celkom jasná.
- Napríklad Funkcionálne programovanie je väčšinou zaradené medzi deklaratívne paradigmy. Napriek tomu, mnoho kódu ktorý sa označuje ako "funkcionálny" je skôr imperatívneho charakteru (čo je priamy protipól deklaratívnosti).

Programovacie paradigmy

- Deklaratívne programovanie - popisujeme logiku výpočtu
- Imperatívne programovanie - popisujeme tok výpočtu

Pri klasickom programovaní výrazne precažuje imperatívne programovanie.

Deklaratívne programovanie

- *select name from users where id=50*
- HTML
- Make (čiastočne)
- Regulárne výrazy
- Constraint programming

Deklaratívne programovanie

- Vyžaduje niečo, čo k logike doplní tok výpočtu.
- Ťažké vo všeobecnosti, často to môže byť výhodné pri aplikácii na úžší problém (Doménovo špecifické jazyky)

Najrozšírenejšie paradigmy

- Procedural programming
- Object-oriented programming
- Functional programming

Plus ďalšie vybrané:

- Event-driven programming
- Aspect-oriented programming
- Generic programming

Procedurálne programovanie

- Procedúry vykonávajú operácie nad dátami (ktoré sú na rôznych úrovniach - globálne, lokálne)

Príklad - Server pri 3 rámci 3-Tier architecture:

- server postupne vykonává DB tranzakciu.

Objektovo orientované programovanie

Základné princípy:

- Abstraction
- Encapsularion
- Polymorphysm
- inheritance

Príklad - Server pri 3 rámci 3-Tier architecture:

- Potrebné ORM - rozšírenie pobjektov za transaction boundary vykonáva tranzakciu (okamžite, alebo "bufferuje").

Objektovo orientované programovanie - viac hardcore prístup

- Podľa mnohých pôvodcov myšlienky OOP napr. Java a C++ nie sú OOP
- Ruby flavoured pravidlá Sandi Metz' Rules For Developers

Funkcionálne programovanie

Základné princípy:

- First class funkcie
- Funkcie vyššieho rádu (vrátane map, filter, accumulate, . . .)
- Pure fukcie
- Referential transparency - Immutable data
- Často silné typovacie systémy
- . . .

Príklad - Server pri 3 rámci 3-Tier architecture:

- Pure funkcie vrátia db procedúru.

Príklad - počítadlo

Procedrálne:

```
a=0
def pocitaj():
    a+=1
    return a
```

Príklad - počítadlo

OOP:

```
class Pocitadlo:
    def __init__(self):
        self.a=0
    def pocitaj(self):
        self.a+=1
        return self
```

Príklad - počítadlo

Funkcionálne bez stavu:

```
class Pocitadlo:
    def __init__(self, pocet=0):
        self.a=pocet
    def pocet(self):
        return self.a

def pocitaj(poc): %toto moze byt aj metoda
    return Pocitadlo(poc.pocet()+1)
```

Príklad - počítadlo

Funkcionálne so stavom (uzáverom):

```
def daj_pocitadlo():
    j=0
    def function_to_return():
        nonlocal j
        j+=1
        return j
    return function_to_return

pocidatlo=daj_pocitadlo()
a=pocitadlo() %1
b=pocitadlo() %2
c=pocitadlo() %3
```

Čo nájdeme vo funkcionálônych jazykoch

Funkcionálne so stavom (uzáverom):

```
def daj_pocitadlo():
    j=0
    def function_to_return():
        nonlocal j
        j+=1
        return j
    return function_to_return

pocidatlo=daj_pocitadlo()
a=pocitadlo() %1
b=pocitadlo() %2
c=pocitadlo() %3
```

Čo nájdeme vo funkcionálônych jazykoch

Ďalší random functional stuff

```
fib = -> n { (n == 0 || n == 1) ? n :
                fib[n - 1] + fib[n - 2] }

s(x) = (1 to x) |> filter (x => x % 2 == 0)
                |> map (x => x * 2)

my_map_and_filter = filter(x => x % 2 == 0)
          . map (x => x * 2)
```

Event-driven programming

GUI, Javascript (príklad promise chaining)

```
new Promise(function(resolve, reject) {
  setTimeout(() => resolve(1), 1000);
}).then(function(result) {
  return result * 2;
}).then(function(result) {
  return result * 2;
}).then(function(result) {
  return result * 2;
});
```

Generic programming

C++, GP funguje s FP aj OOP:

```
template<class Strategy>
class MyCounter {
    int c=0;
    Strategy s;
public:
    MyClass(Strategy s_): s(s_) {}
    int get_value() {return c;}
    int increment() {c=s(c);}
}

increment_by_one=[](int x) {return x+1;}

auto counter=MyCounter(increment_by_one);
```

Zdroje

- GeeksForGeeks: Introduction of Programming Paradigms
- Ray Toal: Programming Paradigms
- Wikipédia: Procedural programming
- javascript.info: Promise-chaining