За Ruby

Чудите се защо Ruby е толкова популярен и обичан език за програмиране? Рубистите го наричат красив и артистичен и въпреки това той е практичен и мощен. Но защо?

Принципи и идеали на създателя на Ruby

Ruby е перфектно балансиран език за програмиране. Създателят муYukihiro „Matz“ Matsumoto е черпил вдъхновение от любимите си езици (Perl, Smalltalk, Eiffel, Ada, и Lisp) докато е писал творението си.

Той често казва, че се е опитал да създаде „естествен, а не лесен за писане език“.

Надграждайки това, той добавя:

Ruby изглежда лесен на повърхността, но всъщност е много сложен, също като човешкото тяло1.

За растежа на Ruby

От първата си публична поява през 1995 година, Ruby привлича отдадени програмисти по целия свят. През 2006, Ruby достигна масов прием.

Ruby-Talk – основният пощенски списък за дискусии относно Ruby достигна 200 съобщения на ден.

Индексът в TIOBE, който измерва растежа на езиците за програмиране, постави Ruby на 9-то място в световен мащаб. Голям принос за този удивителен скок има Ruby on Rails, популярен framework за изграждането на интернет приложения.

Ruby е абсолютно безплатен за употреба, модификация и разпространение.

Философията „всичко е обект“

Първоначално Matz преглежда другите езици в търсене на идеалния синтаксис. Той споделя: „Исках скриптов език за програмиране, по-мощен от Perl и далеч по-обектноориентиран от Python2.“

В Ruby всичко е обект. Всеки бит от информация и код може да притежава свои атрибути и методи. Пример за това ни дава кодът, който следва (метод, извикан върху число).

```
5.times { print "We *love* Ruby -- it's outrageous!" }
```

В много други езици числата и примитивните типове не са обекти. Ruby е повлиян от Smalltalk и дава възможност за извикването на методи на всичките си типове.

Гъвкавост в Ruby

Ruby е считан за гъвкав език поради факта, че програмистите могат свободно да променят всяка една част от кода. Основни части могат да бъдат премахвани или модифицирани. Силата на езика идва от опита за премахването на ограниченията, познати в другите езици.

Пример за това е добавянето на метода `plus`, който дублира оператора (+) директно във вградения клас `Numeric`.

```
class Numeric def plus(x) self.+(x) end end y = 5.plus 6 # y е равно на 11
```

Операторите в Ruby всъщност представляват синтактична „захар“ за методите. Те също могат да бъдат променени.

Експресивност на блоковете

Блоковете в Ruby са друг пример за невероятната гъвкавост на езика. Всеки програмист може да „прикачи“ closure към метод, описващ начина, по който той ще се изпълнява.
„Затварянето“ (closure) се нарича *блок* и се е превърнало в една от най-популярните функционалности за начинаещите, мигрирали от императивни езици като PHP или Visual Basic.

Появата на блоковете в Ruby е вдъхновена от езиците за функционално програмиране.

```
search_engines = %w[Google Yahoo MSN].map do |engine| "http://www." +
engine.downcase + ".com" end
```

В горния отрязък код блокът е реализиран в `do ... end` конструкцията. Методът `map` се прилага за списъка от думи в блока. Много други методи предлагат използването на блокове за прилагането им към специфичен списък/код.

Ruby и Mixin

За разлика от повечето обектно-ориентирани езици, Ruby не предлага множествено наследяване по стандартен начин. Тук то е реализирано с използването на модули (наричани Категории в Objective-C). Модулите са колекции от методи.

Класовете могат да включват модули, като така те получават достъп до техните методи. За пример може да се даде всеки клас, имплементиращ`each` метода, като това е условието за „наследяване“ на модула`Enumerable`, който добавя известен брой методи за работа с `each`.

```
class MyArray include Enumerable end
```

Като цяло рубистите смятат, че това е по-изчистен начин за реализиране на идеята за множествено наследяване, което по-принцип е сложно и ограничаващо в много от случаите.

Визуално представяне на Ruby код

Ruby не се нуждае от декларация на променливи. За сметка на това притежава лесен начин за именуване на различните променливи.

- `var` е локална променлива.
- `@var` е променлива на инстанцията.
- `$var` е глобална променлива.

Тези означения спомагат за четливост, като позволяват на програмиста да идентифицира лесно ролята на всяка променлива. По този начин се премахва употребата на `self.` като идентификатор на всеки член на инстанцията.

Отвъд основните принципи

Ruby притежава множество други черти, като някои от тях са:

- Прихващане и обработка на exceptions, както в Java или Python, което улеснява манипулацията на грешки.
- Притежава истински Garbage Collector за всички обекти, което премахва нуждата от ръчно заделяне и освобождаване на памет.
- Писането на C разширения в Ruby е по-лесно от това на Perl или Python, с елегантно API за извикване на Ruby от C. Това спомага за вграждането на Ruby като скриптов език. Налична е поддръжка на SWIG интерфейс.
- Позволява динамично зареждане на разширителни външни библиотеки в случай, че операционната система го позволява.
- Позволява програмиране с нишки, независимо от операционната система.
- Ruby е преносим език: създаден предимно на GNU/Linux, но работи под много UNIX-базирани операционни системи, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2 и др.