Ruby áttekintés, Rails bevezetés

Kovács Gábor

kovacsg@tmit.bme.hu

BME-TMIT

Ruby

- Általános célú
- Értelmezett, nincs fordítás
- Teljesen objektum-orientált

Ruby összefoglaló 1 – Szintaxis

- Minden (még a zárójelek is) elhagyható, amíg az nem sérti az értelmezést
- Utasítások `;`-vel vagy soremeléssel elválasztva
- Utasítások kapcsos zárójelekkel vagy `do-end`, illetve `begin-end` párokkal blokkokba rendezhetők
- Az értelmezett utasítás vagy blokk visszatérési értéke az utolsó utasítás értéke
- Kisbetű és nagybetű megkülönböztetve
- Az azonosítók referenciák egy objektumra

Ruby összefoglaló 2 – Lexikai elemek

- Kommentek (`#` vagy `=begin-=end`)
- Literálok: egész, lebegőpontos, string (`'` vagy `"`), reguláris kifejezés, tartomány, tömb, hash, lambda, kétértékű és Ruby 2.1-től racionális és komplex
- Azonosítók:
  - Változó: lokális, példány `@`, osztály `@@`, globális `$`, konstans (nagybetűvel kezdődik),
  - Függvény: boolean visszatérési értékű `?`-re végződik, az azonosított objektum értékét megváltoztató `!`, setter `=`-re végződik
- Kulcsszavak
- Üres karakterek: metódus azonosító és a nyitó zárójel közötti üres karakter

Ruby összefoglaló 3 – Adattípusok

- Egész számok
- Lebegőpontos számok
- Stringek
- Tömb (objektumok vesszővel elválasztott listája szögletes zárójelek között)
- Hash (`kulcs => ertek` párok vesszővel elválasztott listája kapcsos zárójelek között)
- Tartomány (`..` és `...`)
- Szimbólum: a programban használt azonosítók, illetve saját azonosítók, mint például kulcsok egy hash-ben. Például `:sym`
- Boolean: `true` és `false`
- A null érték: `nil`
- Az aktuális objektum: `self`

Ruby összefoglaló 4 – Objektumok

- Minden érték objektum, még a primitív típusok is
- Minden objektumot referenciaként érünk el, adunk át
- Az objektum életciklusa a `new` metódus meghívásával kezdődik, a beépített típusok esetén erre nincs külön szükség
- A nem elérhető objektumokat a szemétgyűjtő eljárás távolítja el a memóriából
- Objektum azonosítása: `respond_to?`
- Objektumok azonosságának vizsgálata: érték szerint (`==` operátor), referencia szerint (`equal?` metódus)
- Explicit típuskonverzió: `to_s`, `to_i`, `to_f`
- Implicit típuskonverzió: `to_str`, `to_int`, `to_hash`
- A `nil` `false`-ként viselkedik boolean kifejezésekben

Ruby összefoglaló 5 – Kifejezések, operátorok, változók, metódusok

- Operátorok: a készlet valamivel bővebb, mint C-ben, a precedenciájuk azonos
- A lokális és példányváltozók kezdeti értéke `nil`, inicializálatlan osztály vagy globális változóhoz történő hozzáférés hibát ad
- Változók és metódusok: hivatkozás a `.` vagy a `::` operátorral, ha nincs megadva, akkor a `self` objetumra hivatkozik
- Az operátorok is metódusként definiálhatók és metódusként viselkednek

Ruby összefoglaló 6 – Vezérlési szerkezetek

- Feltételes: `if expr1 then kod {elsif expr2 then kod}* [else kod] end`, `expr` után `;`, soremelés vagy `then` kell, hogy álljon
- Őrfeltétel egysoros kifejezésben: `kod if expr`
- `unless`: akkor hajtódik végre a blokk, ha a kifejezés `false` vagy `nil`
- Többszörös: `case when expr1 kod {when expri kod}+ [else kod] end`
- `while`/`until` ciklus: `while expr blokk end`
- A `while` ciklus addig hajtódik végre, amíg a kifejezés `true`, az `until` ciklus addig hajtódik végre, amíg a kifejezés `true`-vá nem válik
- Egysoros kifejezésben: `blokk while expr`
- `for` ciklus: `for var in enumeration do blokk end`
- Vezérlési folyam módosítása: `return` – visszatérés a metódusból, `break` – kilépés a ciklusból, `next` – következő iteráció, `redo` – iteráció ismétlése, `retry` – a blokk újra-kiértékelése, `throw-catch` – kilépés belső ciklusból, `raise-rescue` – kivételkezelés

Ruby összefoglaló 7 – Metódus, blokk, procedúra mint paraméter

- Metódus definíció:

```
def nev(paramlista); blokk; end
```

```
def m(a)
  yield a*2
end

m('hello') {|l| print "#{l}" }
```

Ruby összefoglaló 8 – Metódus paraméterek, speciális esetek

```
def m(a)
 m=a[:m] || 1
 l=a[:l] || 0
 n=a[:n] || 0
 n+m+l
end
m :n => 3, :l => 4
def m(*varargs)
end
*a = 1,2
a = [1,2]
def m(&block)
  block
end
m { "a" }
```

Ruby összefoglaló 9 – Osztályok

- Definíció: `class Nev; end`
- Már definiált osztálynév használata = viselkedés felüldefiniálása, bővítése
- Példányosítás: `Nev.new`
- Konstruktor: `initialize` metódus definíciója
- Példányváltozók: közvetve definiáljuk azáltal, hogy az osztály egy metódusában használjuk az azonosítóját, alapértelmezés szerint nem hozzáférhető kívülről
- Setter/getter: `def x=(v); @x=v; end`, illetve `def x; @x; end`
- Setter/getter automatikus létrehozása: `attr_accessor :x`, `attr_reader :x`, `attr_writer :x`

Ruby összefoglaló 10 – Osztályok

- Osztálymetódus:
  `class Nev; def Nev.m; print 'm'; end; end`
- Osztályváltozó: `class Nev; @@v=1; end`
- Láthatóság: `public`, `protected`, `private`
- Leszármaztatás: `class B < A`, az ősosztály osztályváltozói, konstansai, osztálymetódusai public és protected metódusai elérhetők, felüldefiniálhatók
- A példányváltozók függetlenek a leszármaztatástól, azok első használatkor jönnek létre
- Referencia a felüldefiniált definiált metódusra: `super`
- Szingleton metódus, csak az adott objektumra érvényes

Ruby összefoglaló 11 – Modulok, mixinek

- Modul
  - A modul osztályok és modulok gyűjteménye
  - Névtér
- Mixin
  - A mixin funkciók gyűjteménye, egy ősosztály
  - A funkcionalitását átörökítheti egy leszármazott osztályba
  - Azonban a leszármazottban figyelmen kívül hagyjuk az *is-a* relációt
  - Ruby-ban modullal valósítható meg, az `include` vagy az `extend` kulcsszóval tehető elérhetővé a funkcionalitása az osztályokban
- Metódus keresési sorrend: szingleton, példány, modul, ősosztályok

Ruby összefoglaló 12 – Modulok, mixinek

```
module M; end
class A
  include M
end
class B < A; end
class C < B; end
b = B.new
b.instance_of? A   #=> false
b.instance_of? B   #=> true
b.instance_of? C   #=> false
b.instance_of? M   #=> false
b.kind_of? A       #=> true
b.kind_of? B       #=> true
b.kind_of? C       #=> false
b.kind_of? M       #=> true
```

Ruby Gem

- Csomagkezelő rendszer Ruby függvénykönyvtárak kezelésére: telepítés, eltávolítás, függőségek
- Gem: alkalmazás vagy függvénykönyvtár
- A telepített függvénykönyvtár használható `require` után
- Rails telepítése: `gem install rails`
- A szükséges függvénykönyvtárak telepítése: `bundle install`

YAML 1

- YAML: adatszerializációra alkalmas nyelv
- Három típus: skalár, sorozat, map
- Adattípusok: `1` egész, `1.1` lebegőpontos, `"String"`, `Yes` boole
- Szintakszis:
  - `kulcs: ertek`: asszociatív tömb
  - `{kulcs1: ertek1, kulcs2: ertek2}`: asszociatív tömb másképpen
  - `-` : lista elemei
  - `[elem1, elem2]`: lista másképpen
  - `#`: komment
  - `---`: dokumentum kezdete egy fájlban
  - `...`: dokumentum vége egy fájlban

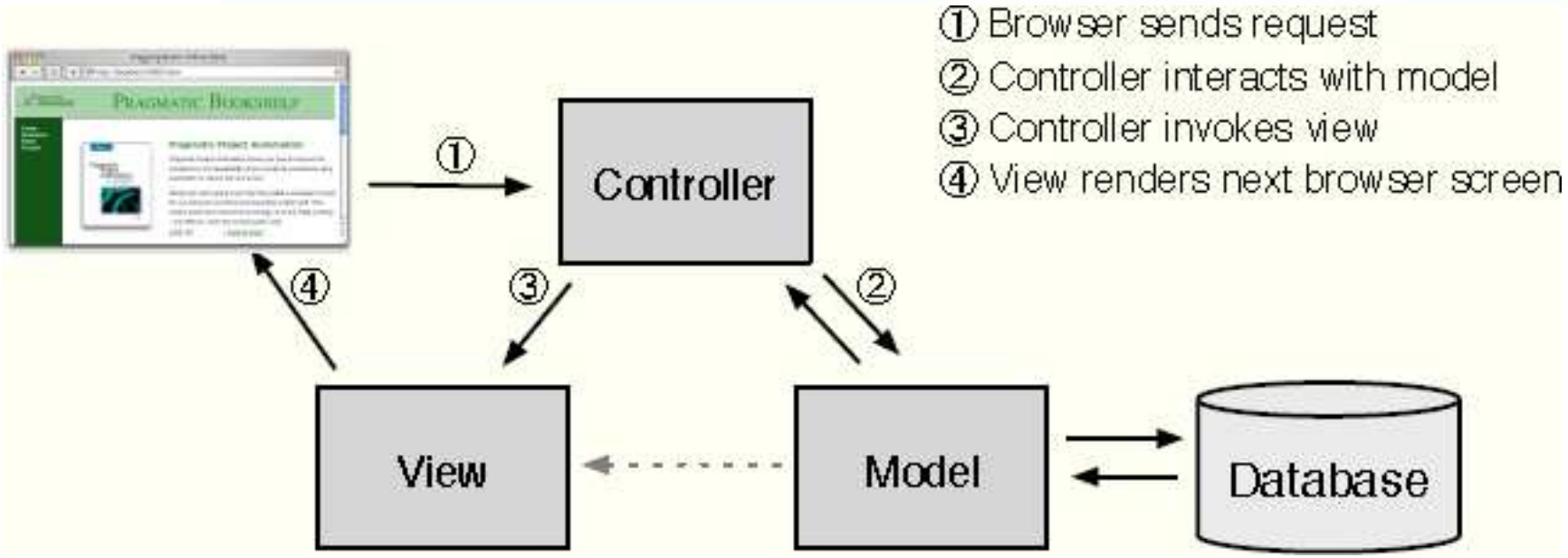
YAML 2

```
# SQLite version 3.x
#   gem install sqlite3-ruby
development:
  adapter: sqlite3
  database: db/development.sqlite3
  pool: 5
  timeout: 5000
```

Rails alapelvek

- eXtreme Programming
- MVC
- RESTful HTTP
- DRY – Don’t repeat yourself
- Kód konvenciók XML konfiguráció helyett

MVC


① Browser sends request
② Controller interacts with model
③ Controller invokes view
④ View renders next browser screen
Controller
View
Model
Database
①
②
③
④

SQLite

- Beágyazott SQL adatbáziskezelő
- Beágyazott = nincs szerver, az alkalmazáson belül fut
- SQL-92
- Tranzakciók támogatása
- Dinamikus típusok: a típus az értékhez tartozik, nem a konténerhez (attribútum); típusok: NULL, INTEGER, REAL, TEXT, BLOB
- Egy fájlban használja a táblák tárolására
- Ideális kisforgalmú weboldalakhoz, beágyazott rendszereken
- Nem jó választás nagyforgalmú weboldalakhoz és kliens-szerver architektúrájú rendszerekhez

Webszerverek

- Webrick
  - Egyszerű HTTP szerver
  - Egy Ruby függvénykönyvtár
- Mongrel, Puma, Unicorn
  - Fejlettebb HTTP szerver
  - Proxy és terheléselosztás funkciókkal
- Apache, Nginx
  - Phusion Passenger plugin, `mod_rails` Apache2-höz és nginx-hez

Rails komponensek 1 – ActiveRecord

- `ActiveRecord` Ruby modul
- Modell a Rails MVC architektúrájában
- Az adatainkat egy relációs adatbázisban tároljuk
- Object-Relational Mapping, elfedi az adatbázis kapcsolatot
  - Adatbázis tábla `orders` és `Order` osztály
  - Egy rekord a táblában egy példány
  - Egy tábla attribútum egy osztály attribútum
  - Az id
- Táblák közötti reláció
- Adatvalidáció, életciklus monitorozás és visszahívás, tranzakciók

Rails komponensek 2 – ActionPack

- A Rails MVC architektúrájában a View és a Controller, szorosan összefonódtak
- ActionPack: `ActionView` és `ActionController` Ruby modulok
- `ActionView`: Ruby beágyazása HTML-be, XML-be, JavaScript-be
- Kontroller:
  - Kapcsolatot teremt a HTTP kérések és az akciók között
  - Kérés paramétereinek kinyerése
  - Karbantartja a helper modulokat
  - Sütik, session karbantartása
  - Flash – egymást követő felhasználói akciók közötti üzenetátadás
  - Gyorsítótár-kezelés

Rails komponensek 3

- `ActionMailer`: email támogatás
- `ActiveResource`: REST támogatás
- `ActiveModel`: `ActiveRecord` viselkedés fapados Ruby osztályok számára
- `ActiveSupport`: segédosztályok
  - Szerializáció: pl. YAML
  - Kiegészítések az alaptípusokhoz
  - Unicode
- `ActiveJob`: aszinkron feladatok

Rails alkalmazás környezetek

- A Rail három környezetet definiál:
  - development: a fejlesztői gépen használt környezet
  - test: a tesztek futtatására használt környezet
  - production: az éles rendszer
- Mindhárom saját adatbázissal rendelkezik

Rails alkalmazások szerkezete 1

- Ami egy új Rail alkalmazás létrehozásakor keletkezik:
  - `Gemfile`: gem függőségek definíciója
  - `Rakefile`: `rake` célok
  - `app/`: alkalmazás specifikus kódok
  - `app/assets`: a weboldal statikus erőforrásai, képek, javascriptek, stíluslapok
  - `app/channels`: websocket kapcsolatok
  - `app/controllers/`: a konroller osztályokat (`ApplicationController` leszármazottak) tartalmazó könyvtár
  - `app/helpers/`: hasznos (általában nézetek számára HTML kódot generáló) függvényeket tartalmazó modulok
  - `app/javascript`: Node.js integráció
  - `app/jobs`: aszinkron műveletek
  - `app/mailers/`: levélküldő osztályok
  - `app/models/`: model osztályokat tartalmazó könyvtár
  - `app/views/`: a beágyazott Ruby kódot tartalmazó HTML oldalakat és layoutokat tartalmazó könyvtár

Rails alkalmazások szerkezete 2

- `bin/`: parancsfájlok automatikus kódgeneráláshoz, futtatáshoz
- `config/`: környezetek, lokalizáció, adatbáziskapcsolatok, URL leképezések
- `db/`: aktuális adatbázis, migrációk, kezdeti értékek
- `lib/`: alkalmazásspecifikus függvénykönyvtárak
- `log/`: napló
- `public/`: a webszerver területe, statikus oldalak
- `test/`: funkcionális, integrációs, teljesítmény- és egységtesztek moduljai, tesztadatok
- `tmp/`: session, cache
- `vendor/`: egyéb függvénykönyvtárak (pl. korábbi verzió egy gem-ből)