



# ***Ruby áttekintés, Rails bevezetés***

Kovács Gábor

`kovacsg@tmit.bme.hu`

BME-TMIT

- ⑥ Általános célú
- ⑥ Értelmezett, nincs fordítás
- ⑥ Teljesen objektum-orientált

# Ruby összefoglaló 1 – Szintaxis

- ⑥ Minden (még a zárójelek is) elhagyható, amíg az nem sérti az értelmezést
- ⑥ Utasítások ; -vel vagy sosemeléssel elválasztva
- ⑥ Utasítások kapcsos zárójelekkel vagy `do-end`, illetve `begin-end` párokkal blokkokba rendezhetők
- ⑥ Az értelmezett utasítás vagy blokk visszatérési értéke az utolsó utasítás értéke
- ⑥ Kisbetű és nagybetű megkülönböztetve
- ⑥ Az azonosítók referenciák egy objektumra

# Ruby összefoglaló 2 – Lexikai elemek

- ⑥ Kommentek (`#` vagy `=begin-=-end`)
- ⑥ Literálok: egész, lebegőpontos, string (`'` vagy `"`), reguláris kifejezés, tartomány, tömb, hash, lambda, kétértékű és Ruby 2.1-től racionális és komplex
- ⑥ Azonosítók:
  - △ Változó: lokális, példány `@`, osztály `@@`, globális `$`, konstans (nagybetűvel kezdődik),
  - △ Függvény: boolean visszatérési értékű `?`-re végződik, az azonosított objektum értékét megváltoztató `!`, setter `=`-re végződik
- ⑥ Kulcsszavak
- ⑥ Üres karakterek: metódus azonosító és a nyitó zárójel közötti üres karakter

# Ruby összefoglaló 3 – Adattípusok

- ⑥ Egész számok
- ⑥ Lebegőpontos számok
- ⑥ Stringek
- ⑥ Tömb (objektumok vesszővel elválasztott listája szögletes zárójelek között)
- ⑥ Hash (kulcs => ertek párok vesszővel elválasztott listája kapcsos zárójelek között)
- ⑥ Tartomány (... és ...)
- ⑥ Szimbólum: a programban használt azonosítók, illetve saját azonosítók, mint például kulcsok egy hash-ben. Például :sym
- ⑥ Boolean: true és false
- ⑥ A null érték: nil
- ⑥ Az aktuális objektum: self

# Ruby összefoglaló 4 – Objektumok

- ⑥ Minden érték objektum, még a primitív típusok is
- ⑥ Minden objektumot referenciaként érünk el, adunk át
- ⑥ Az objektum élelciklusa a `new` metódus meghívásával kezdődik, a beépített típusok esetén erre nincs külön szükség
- ⑥ A nem elérhető objektumokat a szemétyűjtő eljárás távolítja el a memóriából
- ⑥ Objektum azonosítása: `respond_to?`
- ⑥ Objektumok azonosságának vizsgálata: érték szerint (`==` operátor), referencia szerint (`equal?` metódus)
- ⑥ Explicit típuskonverzió: `to_s`, `to_i`, `to_f`
- ⑥ Implicit típuskonverzió: `to_str`, `to_int`, `to_hash`
- ⑥ A `nil` `false`-ként viselkedik boolean kifejezésekben

# Ruby összefoglaló 5 – Kifejezések, operátorok, változók, metódusok

- ⑥ Operátorok: a készlet valamivel bővebb, mint C-ben, a precedenciájuk azonos
- ⑥ A lokális és példányváltozók kezdeti értéke `nil`, inicializálatlan osztály vagy globális változóhoz történő hozzáférés hibát ad
- ⑥ Változók és metódusok: hivatkozás a `.` vagy a `::` operátorral, ha nincs megadva, akkor a `self` objektumra hivatkozik
- ⑥ Az operátorok is metódusként definiálhatók és metódusként viselkednek

# Ruby összefoglaló 6 – Vezérlési szerkezetek

- ⑥ **Feltételes:** `if expr1 then kod {elsif expr2 then kod}* [else kod] end, expr` után `;`, soremelés vagy `then` kell, hogy álljon
- ⑥ **Órfeltétel egysoros kifejezésben:** `kod if expr`
- ⑥ **unless:** akkor hajtódik végre a blokk, ha a kifejezés `false` vagy `nil`
- ⑥ **Többszörös:** `case when expr1 kod {when expri kod}+ [else kod] end`
- ⑥ **while/until ciklus:** `while expr blokk end`
- ⑥ **A while ciklus** addig hajtódik végre, amíg a kifejezés `true`, **az until ciklus** addig hajtódik végre, amíg a kifejezés `true`-vá nem válik
- ⑥ **Egysoros kifejezésben:** `blokk while expr`
- ⑥ **for ciklus:** `for var in enumeration do blokk end`
- ⑥ **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

## 6 Metódus definíció:

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

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

```
m('hello') {|| 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

- ⑥ Definíció: `class Nev; end`
- ⑥ Már definiált osztálynév használata = viselkedés felüldefiniálása, bővítése
- ⑥ Példányosítás: `Nev.new`
- ⑥ Konstruktor: `initialize` metódus definíciója
- ⑥ Példányváltozók: közvetve definiáljuk azáltal, hogy az osztály egy metódusában használjuk az azonosítóját, alapértelmezés szerint nem hozzáférhető kívülről
- ⑥ Setter/getter: `def x=(v); @x=v; end, illetve def x; @x; end`
- ⑥ 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 őssztály 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`

## Singleton metódus, csak az adott objektumra érvényes

# Ruby összefoglaló 11 – Modulok, mixinek

## ⑥ Modul

- △ A modul osztályok és modulok gyűjteménye
- △ Névtér

## ⑥ Mixin

- △ A mixin funkciók gyűjteménye, egy őszosztály
- △ 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

## ⑥ Metódus keresési sorrend: singleton, példány, modul, őszosztályok

# 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

- ⑥ Csomagkezelő rendszer Ruby függvénykönyvtárak kezelésére: telepítés, eltávolítás, függőségek
- ⑥ Gem: alkalmazás vagy függvénykönyvtár
- ⑥ A telepített függvénykönyvtár használható `require` után
- ⑥ Rails telepítése: `gem install rails`
- ⑥ A szükséges függvénykönyvtárak telepítése: `bundle install`

# YAML 1

- ⑥ YAML: adatserializációra alkalmas nyelv
- ⑥ Három típus: skalár, sorozat, map
- ⑥ Adattípusok: 1 egész, 1.1 lebegőpontos, "String", Yes boole
- ⑥ Szintakszis:
  - △ kulcs: érték: asszociatív tömb
  - △ {kulcs1: érték1, kulcs2: érték2}: asszociatív tömb másképpen
  - △ - : lista elemei
  - △ [elem1, elem2]: lista másképpen
  - △ #: komment
  - △ ---: dokumentum kezdete egy fájlban
  - △ ...: dokumentum vége egy fájlban



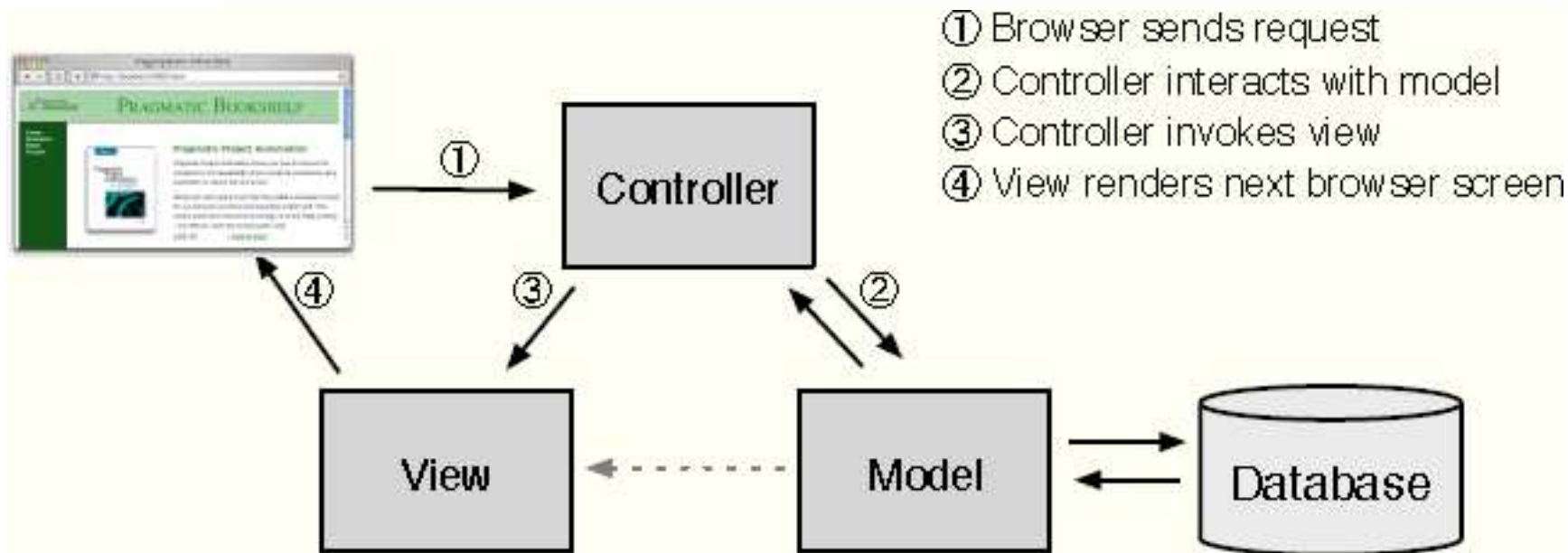
## YAML 2

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

# ***Rails alapelvek***

- ⑥ eXtreme Programming
- ⑥ MVC
- ⑥ RESTful HTTP
- ⑥ DRY – Don't repeat yourself
- ⑥ Kód konvenciók XML konfiguráció helyett

# MVC



# SQLite

- ⑥ Beágyazott SQL adatbáziskezelő
- ⑥ Beágyazott = nincs szerver, az alkalmazáson belül fut
- ⑥ SQL-92
- ⑥ Tranzakciók támogatása
- ⑥ Dinamikus típusok: a típus az értékhez tartozik, nem a konténerhez (attribútum); típusok: NULL, INTEGER, REAL, TEXT, BLOB
- ⑥ Egy fájlban használja a táblák tárolására
- ⑥ Ideális kisforgalmú weboldalakhoz, beágyazott rendszereken
- ⑥ Nem jó választás nagyforgalmú weboldalakhoz és kliens-szerver architektúrájú rendszerekhez

# Webszerverek

## ⑥ Webrick

- △ Egyszerű HTTP szerver
- △ Egy Ruby függvénykönyvtár

## ⑥ Mongrel, Puma, Unicorn

- △ Fejlettebb HTTP szerver
- △ Proxy és terheléselosztás funkciókkal

## ⑥ Apache, Nginx

- △ Phusion Passenger plugin, `mod_rails` Apache2-höz és nginx-hez

# ***Rails komponensek 1 – ActiveRecord***

- ⑥ ActiveRecord Ruby modul
- ⑥ Modell a Rails MVC architektúrájában
- ⑥ Az adatainkat egy relációs adatbázisban tároljuk
- ⑥ 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
- ⑥ Táblák közötti reláció
- ⑥ Adatvalidáció, élelciklus monitorozás és visszahívás, tranzakciók

# ***Rails komponensek 2 – ActionPack***

- ⑥ A Rails MVC architektúrájában a View és a Controller, szorosan összefonódtak
- ⑥ ActionPack: `ActionView` és `ActionController` Ruby modulok
- ⑥ `ActionView`: Ruby beágyazása HTML-be, XML-be, JavaScript-be
- ⑥ **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***

- ⑥ `ActionMailer`: email támogatás
- ⑥ `ActiveResource`: REST támogatás
- ⑥ `ActiveModel`: `ActiveRecord` viselkedés fapados Ruby osztályok számára
- ⑥ `ActiveSupport`: segédosztályok
  - △ Szerializáció: pl. YAML
  - △ Kiegészítések az alaptípusokhoz
  - △ Unicode
- ⑥ `ActiveJob`: aszinkron feladatok



# *Rails alkalmazás környezetek*

- ⑥ A Rail három környezetet definiál:
  - △ development: a fejlesztői gépen használt környezet
  - △ test: a tesztek futtatására használt környezet
  - △ production: az éles rendszer
- ⑥ Mindhárom saját adatbázissal rendelkezik

# Rails alkalmazások szerkezete 1

## 6 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 webservert területe, statikus oldalak
- test/: funkcionális, integrációs, teljesítmény- és egységtesztek moduljai, tesztadatok
- tmp/: session, cache
- vendor/: egyéb függvénykönyvtárak (pl. korábbi verzió egy gem-ből)