Ruby on Rails – Webprogramozás könnyedén

A Ruby on Rails-t sokan tekintik az internet programozás egyik zászlóshajójának. Vajon miért övezi ekkora hírverés ezt a mindössze két éves keretrendszert?

A *Ruby on Rails*, mindent tartalmaz egy csomagban, amire adatbázis alapú weboldalak kialakításához szükségünk lehet. Az indulás is egyszerű, hiszen telepítés után mindössze az adatbázis kapcsolat paramétereit kell beállítanunk és már fejleszthetünk is. Lehetővé teszi, hogy a fontos dolgokkal foglalkozzunk fejlesztés közben. A kevésbé fontos és ismétlődő feladatokat pedig nyugodtan rábízhatjuk.

Mindenek alapja a Ruby

Rails-al fejleszteni annyit tesz, mint megérteni és megszeretni a filozófiát ami mögötte áll. A keretrendszer alapját adó *Ruby* nyelv *Japánból* indult hódító útjára.(A ruby angolul rubintot jelent.) A *Ruby* teljesen objektum orientált script nyelv. Szintaktikája nagyon elegáns és egyszerű. Sebessége a *python*-nál kicsit lassabb, de bőven megfelelő weboldalak kiszolgálásához. A legcsodálatosabb jellemvonása, hogy segítségével bámulatosan olvasható kódot írhatunk. Álljon itt pár példa!

```
5.times { print "Hello Linux!" }
# Nem tesz mást, mint 5
alkalommal kiírja, hogy
"Helló Linux".

exit unless
➥"Linuxvilág".include? "nux"
# Kilép, ha a "Linuxvilág" nem
tartalmazza a "nux" szöveget.
```

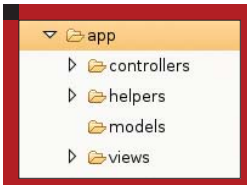
David filozófiája

David Heinemeier Hansson a *Rails* atyja. *David* szerint a bonyolult konfigurációs beállítások, melyek a többi keretrendszerre annyira jellemzőek, csak hátráltatják a munkát. Éppen ezért a *Rails* kerüli ezeket, inkább nagymértékben támaszkodik a konvenciókra. Áthatja a *Rails* működését egy másik fontos elv is, a *DRY* (*„don't repeat yourself"*), ami annyit jelent, hogy „ne ismételd magad hiába"! Ha valamit valahol leírtál, akkor azt soha máshol ne kelljen leírni még egyszer. Ez segít abban, hogy a *Rails*-al öröm legyen a fejlesztés. Eredményül átlátható és könnyen továbbfejleszthető programkódot kapunk.

A keretrendszer

Az elmélet bevezetőn immár túljutottunk, vegyük sorba mit kínál nekünk a *Rails*, mint fejlesztőeszköz! Mint szó volt róla a bevezetőben, mindent tartalmaz, amire egy adatbázis alapú weboldal fejlesztéséhez szükséges lehet. Ez azt jelenti, hogy a segítségével a kérés beérkezésétől kezdve egészen a válasz elküldéséig kontrollálhatjuk a folyamatokat. Kezeli a beérkező kéréseket, segíti az adatbázissal való munkát és rengeteg eszközt ad a kezünkbe a hatékony felhasználói felületeket megtervezéséhez is. Mindezt programozói szemmel nézve bámulatosan hatékonyan és átláthatóan teszi. Egy *Rails-el* készített alkalmazás (weboldal) a *Model-View-Controller* séma alapján épül fel. Vagyis külön tároljuk azokat a kódrészleteket, amelyek az adatokkal foglalkoznak, külön a megjelenítés sablonjait és külön az üzleti logikát. Az üzleti logika az a része a programnak, amely megmondja, hogy ha a felhasználó ezt az címet írta be a böngészőjébe akkor ennek és ennek kell történnie. Vegyük sorra a főbb komponensek működését!

Álljon itt pár olyan weboldal, amerre érdemes körülnézni annak, aki a *Ruby* nyelvet szeretné kicsit közelebbről megismerni:
➲ http://tryruby.hobix.com/
Próbáld ki *Ruby*-t a böngésződdben!
➲ http://poignantguide.net/ruby/
Az egyik legmókásabb programozásról szóló könyv amit életemben láttam.
➲ http://www.ruby-lang.org
A *Ruby* hivatalos oldala.



1. ábra Egy Rails alkalmazás a Model-View-Controller séma alapján épül fel

Az ActiveRecord megszelídíti az adatbázist

Az *ActiveRecord* segítségével játékká egyszerűsödik az adatbázissal való munka. Adatbázis tábláinkat osztályokként kezelhetjük, az adatbázis egy sora pedig egy példánynak felel meg. Ezt hívják szakszóval *Object Relacional Mapper-nek* (*ORM*).
A táblák közötti kapcsolatot rövid, egy soros utasításokkal adhatjuk meg. Ha például minden *user*-hez tartozhat egy cím, akkor azt így írhatjuk:

```
class User < ActiveRecord::Base
  has_one  'address'
end
```

2. ábra Érdeklődj, kezdd el használni, fejlődj és kapcsolódj be a fejlesztésébe! – áll a Rails weboldalán

```
class Address <
➥ActiveRecord::Base
  belongs_to  'user'
end
```

Létrehoztuk tehát az User és az Address osztályokat! Az *ActiveRecord* a konvenciók alapján az *User* osztályhoz az adatbázisban az *users* táblát fogja társítani, az *Address* osztályhoz pedig az *addresses* táblát. Az két osztály közötti kapcsolatot adatbázis szinten az *addresses* táblában található külső kulcs (*foreign key*) oszlop fogja biztosítani. Ezek után már gyerekjáték lesz dolgozni az adatainkkal. Ha például van egy „Attila” nevű felhasználónk, akinek van címe az adatbázisban, azt így módosíthatjuk:

```
user =
➥User.find_by_name('Attila')
#lekérjük az adatbázisból
Attila adatait
user.address = 'itt lakom
➥látod, ez az a hely'
#felülírjuk a kapcsolódó cím
rekordot
```

Ugye milyen egyszerű? Olyan az egész mintha angolul, tőmondatokban írnánk le az utasításainkat. Egyszerűbben már nem is lehetne.

Az ActionController a rendszer lelke

Mi történik akkor amikor egy *Rails* alkalmazás kiszolgál egy oldalt? Ha a felhasználó beírja, hogy „/user/edit/” A *Rails* rögtön az *UserController* osztály *edit* metódusát fogja meghívni. Ha az *edit* metódus nem létezik, akkor pedig megpróbálja megkeresni és a felhasználónak elküldeni a metódushoz tartozó *html* sablont. Ha az *UserController* sem létezik akkor már baj van. Ekkor már egy kövér hibaüzenettel fogunk találkozni. A *Rails*-ben tehát létrehozhatunk osztályokat, melyek a kérések kezelését fogják elvégezni. Ez az a rész, ahol az üzleti logika megbújik.
Az *ActionController* tehát kezeli a bejövő kéréseket, és segít a válasz küldésében is. Ez az a komponens, amelyik eldönti, hogy milyen kérésre mit kell válaszolni, majd pedig a sablonok alapján felépíti a válaszul küldendő oldalt. Itt ugyancsak megfigyelhetőek a *Rails* konvenciói, hiszen a kéréseket lekezelő kódok és a válaszok megjelenítéséért felelős *HTML* sablonok pusztán nevük alapján kapcsolódnak egymáshoz.

És még sok más...

A rendszer két legfontosabb összetevőjét megismertük. Vegyük sorra azokat az eszközöket, melyek segítenek a munka során!
Az egyik például a kódgenerálás. Szinte bármilyen funkciójú kód sablonját legyártathatjuk a keretrendszerrel. Ilyen esetben a *Rails* létrehozza a szükséges fájlokat, bennük pedig elhelyezi azokat az alap kódokat, amelyek segítik az elindulást. Ha például egy új *model*-t szeretnénk létrehozni „User” néven, akkor a következő parancsot kell kiadnunk az alkalmazásunk gyökérkönyvtárában:

```
ruby script/generate model User
```

Ennek hatására a létrejön egy „User” nevű modell, minden hozzá tartozó fájllal együtt. Hasonló módon generálhatunk számos mást is, lényegesen leegyszerűsítve ez által a fejlesztést. A másik érdekes dolog a *scaffold* (magyarul álványzat). Ez az a lehetőség amelybe mindenki beleszeret a kezdetekben. Nevéhez híven megtámogatja a fejlesztést, ugyanis a *scaffold*-dal létrehozhatunk egy alap felületet, melyen keresztül egy *model*-t kezelhetünk. Ez azt jelenti, hogy minden komolyabb programozás nélkül kapunk egy olyan oldalt, amelyen egy adatbázis tábla (ez felel meg egy *model*-nek) tartalmát változtathatjuk. Később, természetesen saját felületet építhetünk az adatok eléréséhez, de a kezdetekben nagyban megkönnyíti a tesztadatok bevitelét.

Összefoglalás

Röviden áttekintettük a *Rails* főbb tulajdonságait. Mivel a legtöbb dolog akkor válik érdekessé amikor munkára fogjuk, így a következő részben, részekben egy alkalmazás fejlesztését fogom bemutatni. Aki nagyon türelmetlen, az látogassa meg a projekt weboldalát és nézze meg a videókat!

A Model-View-Controller séma (MVC)

Az *MVC* egy program tervezési minta. A mintát követő alkalmazásokban szeparálni igyekszünk a különböző funkciókat végző programrészeket. Főleg asztali alkalmazás fejlesztésekor dolgoznak a séma alapján. Az adatokkal dolgozó programrészeket *Model*-nek, a megjelenítés sablonjait *View*-nek, a program lelkét adó, felhasználói interakciót és üzleti logikát rejtő részeket pedig *Controller*-nek nevezzük. Az így megkülönböztetett programrészeket általában különböző könyvtárban is tároljuk. Előnye ennek a tervezési mintának, hogy átlátható, könnyen karbantartható kódot kapunk.

„Rails is the killer app for Ruby.”, vagyis a *Ruby* nyelv számára a *Rails* hozhatja meg a régóta várt ismertséget, írta *Yukihiro Matsumoto*, a nyelv megalkotója. Állítását igazolhatom máris, hiszen a fejlesztői listák tanúsága szerint rengetegen ismerkednek meg a nyelvvel miután felkeltette a kíváncsiságukat a *Rails*. Ez fordítva is igaz, hiszem a *Ruby* nyelv filozófiája nélkül a keretrendszer csak egy lenne a sok közül.

Juhász Attila
(rabszolga@goraffe.hu)
Az Információ Technológiai Kar hallgatója a Pázmány Péter Katolikus Egyetemen. Érdeklődik a bioinformatika és a neurális hálózatok iránt. A fotózás és a tánc mellett öt éve foglalkozik webgrafikákkal. A linux terjesztések közül a Gentoo és az Ubuntu áll legközelebb a szívéhez. Fotós oldala a http://people.goraffe.com/attila címen található.