



Webfejlesztés villámgyorsan Ruby on Rails alapokon

Kovács Gábor

`kovacsg@tmit.bme.hu`

BME-TMIT

A tantárgy adatlapja

- ⑥ 2 kredit
- ⑥ Előadás: páratlan hét kedd 12:15-14:00 I.B.140
- ⑥ Gyakorlat: páros hét kedd 12:15-14:00 I.B.140
- ⑥ Követelmények:
 - △ Hat fejlesztési gyakorlat megoldása – félévközi jegy a teljesített feladatok alapján
 - △ Leadási határidő a következő gyakorlat előtti nap
 - △ Hat feladat megoldása kötelező
 - △ Egy késedelmes, pótlólagos leadása lehetséges a következő leadási határidőig
 - △ Egy feladat pótolható vagy javítható a pótlási héten
- ⑥ A tantárgy weboldala:
<https://twiki.db.bme.hu/twiki/bin/view/Student/Ruby/WebHome>
- ⑥ Feladatok beadása: rorhf@tmit.bme.hu

A tantárgy tematikája röviden

- ⑥ Bevezetés
- ⑥ Szoftvertechnológiai összefoglaló
- ⑥ A Ruby programozási nyelv
- ⑥ Adatbázis, migráció, ORM
- ⑥ Eseménykezelés, nézet, Ruby beágyazás
- ⑥ Validáció, tesztelés
- ⑥ Kiegészítők

Rails filozófia

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

Agilis fejlesztési módszertan 1

Motiváció

- ⑥ Az egyéneket és a személyes kommunikációt a módszertanokkal és eszközökkel szemben
- ⑥ A működő szoftvert az átfogó dokumentációval szemben
- ⑥ A megrendelővel történő együttműködést a szerződéses egyeztetéssel szemben
- ⑥ A változás iránti készséget a tervek szolgai követésével szemben

Agilis fejlesztési módszertan 2

- ⑥ Ügyfél és fejlesztők napi rendszerességű együttműködése, konstans tempójú előrehaladás
- ⑥ Információátadás személyes megbeszélés során, a megrendelő is a helyszínen van
- ⑥ Szoftverváltozat szállítása rövid periódusonként
- ⑥ A működő szoftver az előrehaladás elsődleges mértéke
- ⑥ Korai, illetve folyamatos szállítás a megrendelőnek
- ⑥ Folyamatosan változó követelmények

Agilis fejlesztési módszertan 3



#113 - "AGILE DEVELOPMENT, EXPLAINED" - BY SALVATORE IOVENE, FEB. 21ST 2009

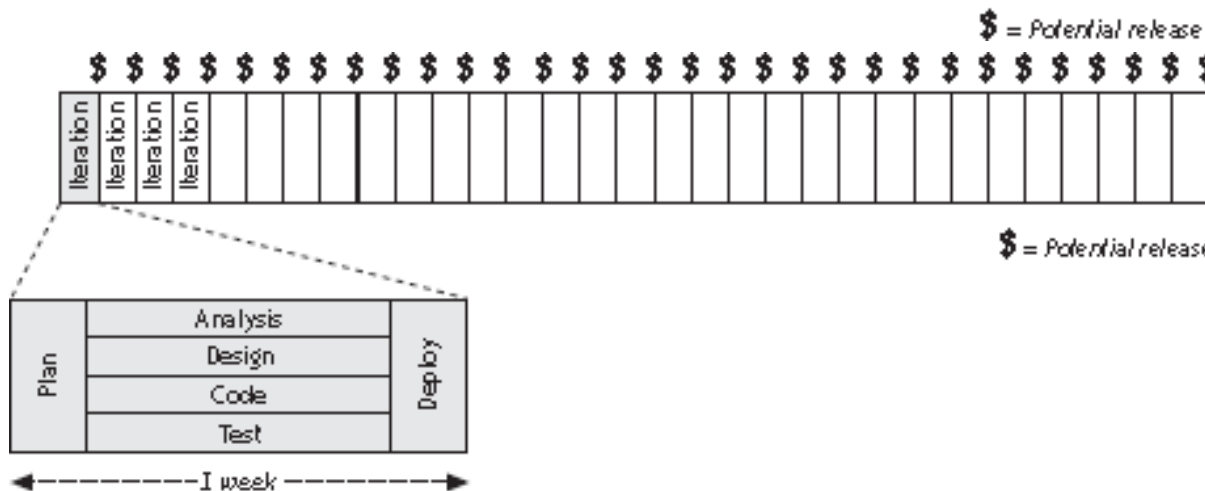
[HTTP://WWW.GEEKHEROCOMIC.COM/](http://www.geekherocomic.com/)

Fejlesztési életciklus modellek

(a) Waterfall lifecycle



(b) Iterative lifecycle



XP élelciklus 1

- ⑥ Vízesés: tervezés, analízis, vázlat, kódolás, tesztelés, telepítés
- ⑥ Iteratív: a vízesés ciklusa néhány alkalommal ismétlődik a projekt során
- ⑥ XP élelciklus: heti rendszerességgel
 - △ Tervezés – üzleti döntések, projektmenedzsment döntések, programozói javaslatok és becslések, megrendelői prioritások
 - △ Elemzés, vázlat, kódolás, tesztelés egyszerre
 - △ Telepítés

XP életciklus 2

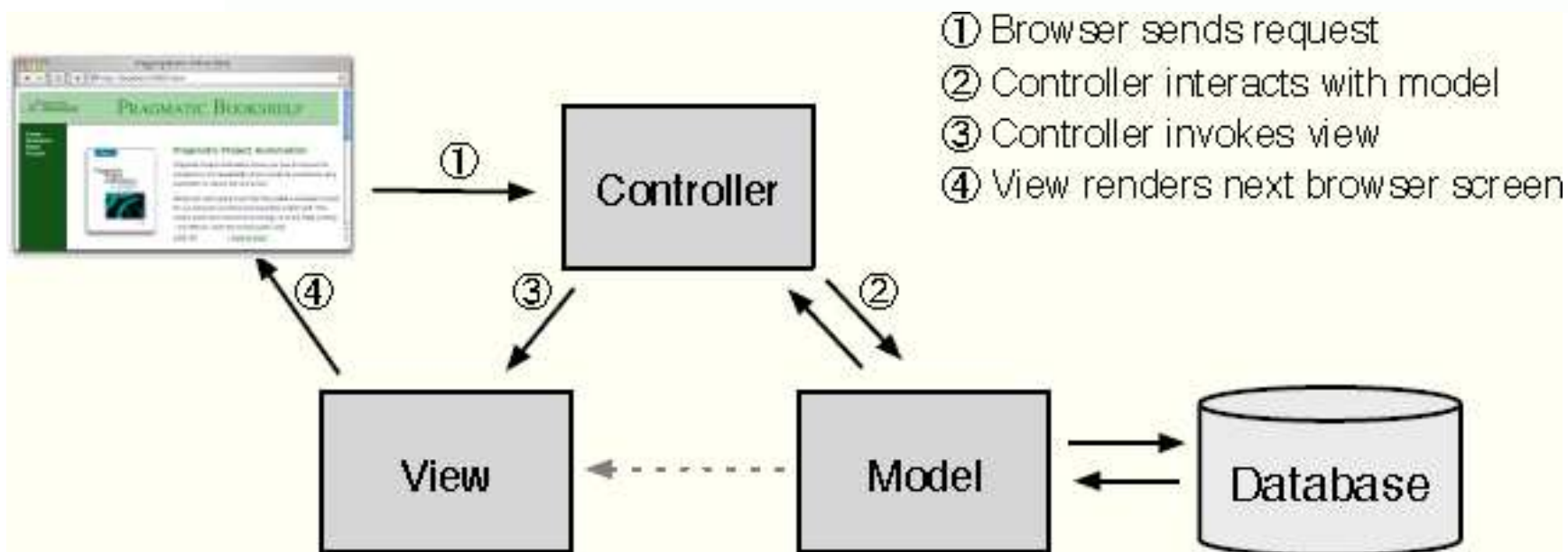
- ⑥ Nem feltétlenül hatékonyabb, viszont rendszeres visszajelzést kap a megrendelőtől
- ⑥ Elemzés = a megrendelő értelmezi a követelményeket a programozók számára, és formalizálja a tesztelők segítségével, UI kinézetet egyeztet a grafikkussal
- ⑥ Vázlat és kódolás = inkrementális, tesztvezérelt fejlesztés apró lépésekben; verziókezelő rendszer és kódolási szabályok használata
- ⑥ Tesztelés = tesztvezérelt fejlesztés azonnali hibajavítással
- ⑥ Telepítés = minden iterációs ciklus végén, demonstráció a megrendelő felé hetente

XP életciklus 3

Tulajdonságok, csapdák

- ⑥ Tervezés történetek alapján (használati esetek szöveggé formázva)
- ⑥ Iterációk: tervezés, kódolás, ellenőrzés, release
- ⑥ Folyamatos refaktorizálás
- ⑥ Gyorsjavítások elkerülendők
- ⑥ Időkeret-kezelés: mindig tovább lehet szépíteni a kódot release előtt; időkeret kutatásra (technológia kiválasztása és megismerése) és megbeszélésekre
- ⑥ Alternatíva választására alkalmas utolsó pillanat azonosításának fontossága
- ⑥ Túlreagált változtatás, a változtatás hatására mindig új probléma merül fel
- ⑥ Rendszereszt helyett a megrendelő tesztel

MVC 1



MVC 2

- ⑥ Model: az alkalmazás/komponens állapotát tárolja és tartja karban
- ⑥ View: a felhasználói felület megjelenítése
- ⑥ Controller: összehangolja az alkalmazást,
 - △ reagál a külső, elsősorban felhasználói inputokra,
 - △ frissíti a modellt,
 - △ és megjeleníti a nézetet

REST 1

- ⑥ REST, REpresentational State Transfer
- ⑥ Alkalmazások közötti kommunikáció
- ⑥ A webszolgáltatások alternatívája
- ⑥ Definíció: a REST elvek halmaza a web szabványok (pl. HTTP, URI) használatáról
 - △ Minden erőforrás rendelkezék azonosítóval
 - △ Az erőforráson elérhető linkek reprezentálják az alkalmazás állapotát
 - △ A HTTP szabványos műveletei mint eljárások
 - △ Erőforrások többféle reprezentációval rendelkeznek
 - △ Állapotmentes kommunikáció

6 Minden erőforrás rendelkezék azonosítóval

- △ Az egyedi azonosító a weben: az URI
- △ Adatbázis id jelenik meg az URI-ban, ez ellentétes az objektumorientált környezetből megszokott adatrejtés filozófiájával
- △ Egy üzleti folyamat vagy annak egy tevékenysége azonosítója is megjelenik az URI-ban
- △ **Például:** `http://www.szervernev.hu/hallgatok/15,`
`http://www.azonlineboltom.hu/rendeles/2016/02/16/5554,`
`http://www.azenvallalatom.hu/fizetesemeles/401`
- △ Az URI azonosíthatja erőforrások csoportját is:
`http://www.szervernev.hu/hallgatok,`
`http://www.azonlineboltom.hu/rendeles/2016/02?fizetes=kartya`

6 Az alkalmazás állapota

- △ Az erőforrás dokumentuma által tartalmazott már erőforrásra mutató linkek határozza meg
- △ Nem egy konkrét, alkalmazásspecifikus adatbázis azonosító jellemzi az oldalt
- △ Állapotváltás = egy link követése
- △ A dokumentum egy-egy linkje mutathat technológiailag más szerverre, más alkalmazásra, üzletileg más cég erőforrására
- △ Példa:

```
<Rendeles megrendelo="http://www.azenvallalom.hu/alkalmazott/401">  
  <Mennyiseg>1</Mennyiseg>  
  <Termek>http://egymasikvallalat.com/termek/1114</Termek>  
  <Megrendelo>http://www.azenvallalom.hu/alkalmazott/401</Megrendelo>  
</Rendeles>
```

6 A kommunikáció állapotmentes, az állapotot a kliens ismeri

HTTP 1

- ⑥ Rövid kitérő: HTTP
- ⑥ Állapotmentes, kérés-válasz protokoll

- ⑥ Kérés:

```
<Muvelet> <Eroforras> <Verzio> CRLF  
[ { <Fejresz_nev>: <Ertek> CRLF }* ]  
CRLF  
[<Torzs>]
```

- ⑥ Válasz:

```
<Verzio> <Statuszkod> <Statusz_uzenet> CRLF  
[ Content-type: <mime_tipus>; [charset=<karakterkod>] CRLF ]  
[ { <Fejresz_nev>: <Ertek> CRLF }* ]  
CRLF  
[<Torzs>]
```

- ⑥ Műveletek: CONNECT, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT, TRACE

HTTP 2

6 Kérés példa:

```
GET / HTTP/1.1
Host: www.tmit.bme.hu
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.19)
Gecko/2010033022 Iceweasel/3.0.14 (Debian-3.0.14-1)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: hu,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

6 Válasz példa:

```
HTTP/1.1 200 OK
Date: Fri, 10 Sep 2010 05:39:31 GMT
Server: Apache
Set-Cookie: PHPSESSID=727e6dac0d03e65621e5adefd452a20e; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0,
pre-check=0
Pragma: no-cache
Keep-Alive: timeout=4, max=8
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
Content-Language: hu
```

REST 4

- ⑥ A HTTP szabványos műveletei mint eljárások az erőforrásokon, minden erőforrás azonos műveletekkel rendelkezik. A műveletek értelmezése
 - △ GET = mutat
 - △ PUT = frissít
 - △ DELETE = töröl vagy elvet
 - △ POST = új vagy hozzáad
- ⑥ Erőforrások többféle reprezentációja:
 - △ Accept fejrész a GET/POST kérésben: pl.
Accept: application/pdf vagy image/jpeg

A Ruby programozási nyelv

A Ruby tulajdonságai

- ⑥ Általános célú
- ⑥ Teljesen objektum-orientált, még primitív típusok sincsenek benne
- ⑥ Értelmező által futtatott, vagyis nincs fordítás
- ⑥ Kizárólag referencia szerinti értékadás van
- ⑥ Paraméterátadás rendezett lista helyett hash halmazzal
- ⑥ Operátorok, mint C-ben
- ⑥ Blokkok sajátos kezelése
- ⑥ Az osztályok és modulok a mixin minta szerint összeszűrhetők
- ⑥ Konvenciók

Hello world

```
irb(main):001:0> puts "Hello, world!"  
Hello, world!  
=> nil
```

Minden objektum:

```
irb(main):002:0> 2.times { puts "Hello, world!".upcase }  
HELLO, WORLD!  
HELLO, WORLD!  
=> 2
```