Webfejlesztés villámgyorsan Ruby on Rails alapokon

Kovács Gábor

kovacsg@tmit.bme.hu

BME-TMIT

A tantárgy adatlapja

- 2 kredit
- Előadás: páratlan hét kedd 12:15-14:00 Q.B.F10
- Gyakorlat: páros hét kedd 12:15-14:00 Q.B.F10
- Követelmények:
  - Hat fejlesztési gyakorlat megoldása – félévközi jegy a teljesített feladatok alapján
  - Leadási határidő a következő gyakorlat előtti nap
  - Hat feladat megoldása kötelező
  - Egy késededelmes, pótlólagos leadása lehetséges a következő leadási határidőig
  - Egy feladat pótolható vagy javítható a pótlási héten
- A tantárgy weboldala:
  `https://twiki.db.bme.hu/twiki/bin/view/Student/Ruby/WebHome`
- Tárggyal kapcsolatos információk: Moodle, Teams
- Feladatok beadása: `rorhf@tmit.bme.hu`

A tantárgy tematikája röviden

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

Rails filozófia

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

Agilis fejlesztési módszertan 1

Motiváció

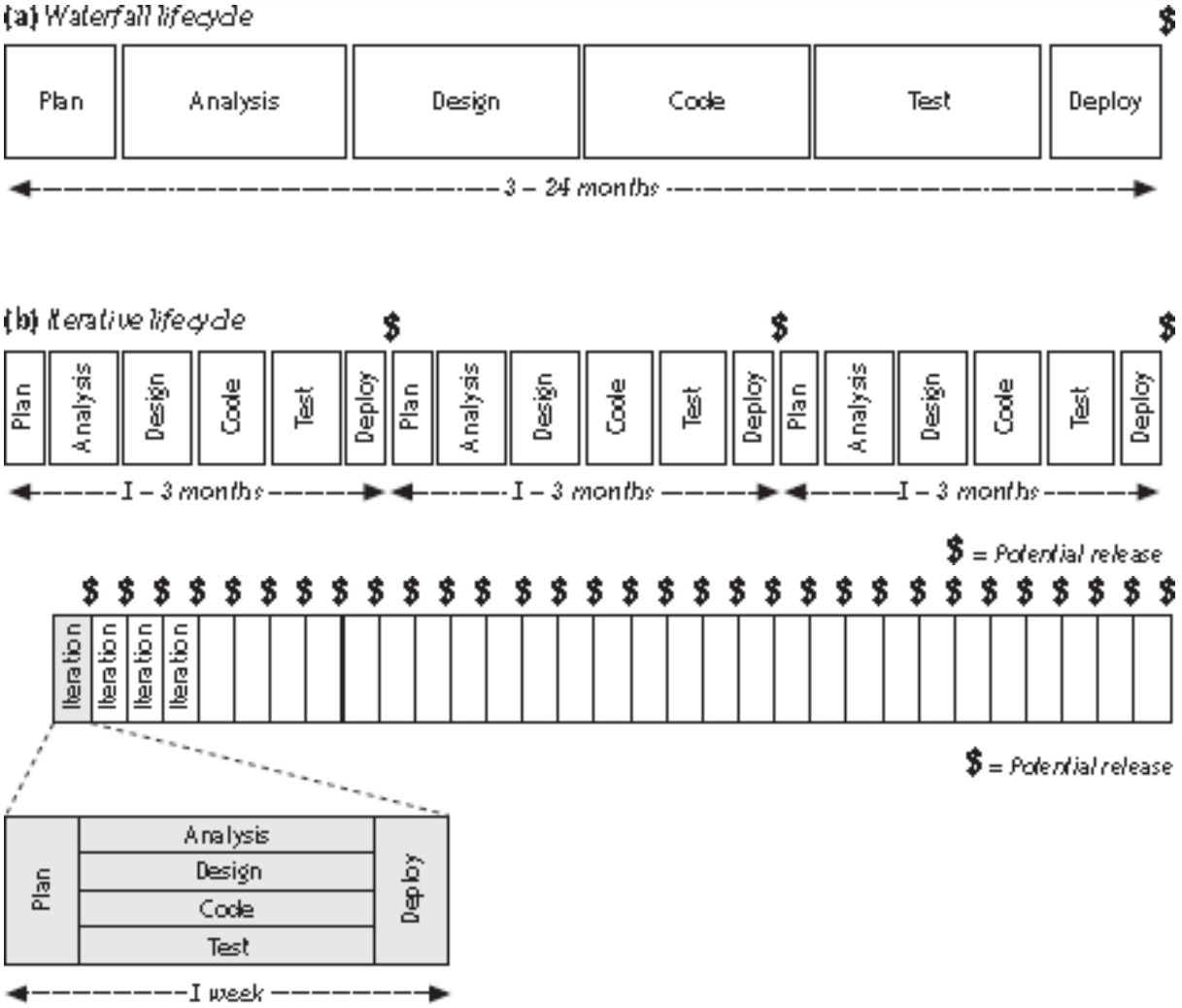
- Az egyéneket és a személyes kommunikációt a módszertanokkal és eszközökkel szemben
- A működő szoftvert az átfogó dokumentációval szemben
- A megrendelővel történő együttműködést a szerződéses egyeztetéssel szemben
- A változás iránti készséget a tervek szolgai követésével szemben

Agilis fejlesztési módszertan 2

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

Agilis fejlesztési módszertan 3

Fejlesztési életciklus modellek



XP életciklus 1

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

XP életciklus 2

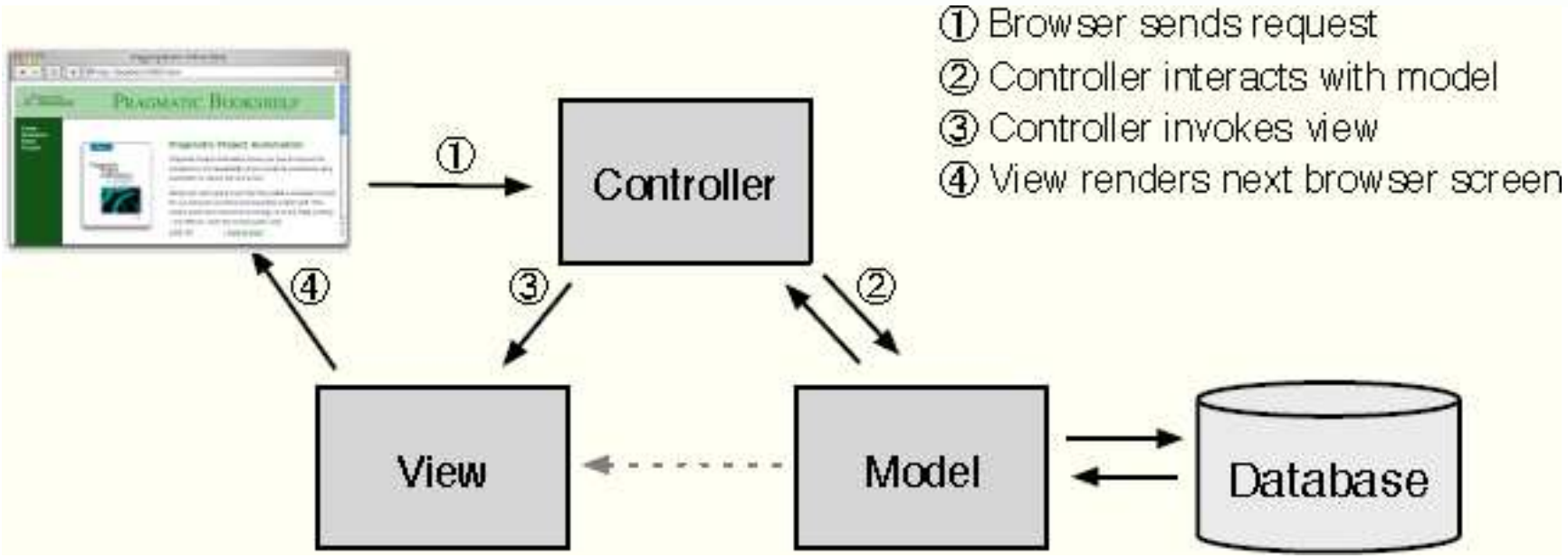
- Nem feltétlenül hatékonyabb, viszont rendszeres visszajelzést kap a megrendelőtől
- Elemzés = a megrendelő értelmezi a követelményeket a programozók számára, és formalizálja a tesztelők segítségével, UI kinézetet egyeztet a grafikussal
- Vázlat és kódolás = inkrementális, tesztvezérelt fejlesztés apró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

XP életciklus 3

Tulajdonságok, csapdák

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

MVC 1


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

MVC 2

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

REST 1

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

REST 2

- Minden erőforrás rendelkezzék 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.azenonlineboltom.hu/rendeles/2020/02/11/5554`,
    `http://www.azenvallalatom.hu/fizetesemeles/401`
  - Az URI azonosíthatja erőforrások csoportját is:
    `http://www.szervernev.hu/hallgatok`,
    `http://www.azenonlineboltom.hu/rendeles/2020/02?fizetes=kartya`

REST 3

- Az alkalmazás állapota
  - Az erőforrás dokumentuma által tartalmazott más 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.azenvallalatom.hu/alkalmazott/401">
    <Mennyiseg>1</Mennyiseg>
    <Termek>http://egymasikvallalat.com/termek/1114</Termek>
    <Megrendelo>http://www.azenvallalatom.hu/alkalmazott/401</Megrendelo>
</Rendeles>
```

- A kommunikáció állapotmentes, az állapotot a kliens ismeri

HTTP 1

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

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

- Válasz:

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

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

HTTP 2

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
```

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

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

A Ruby programozási nyelv

A Ruby tulajdonságai

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