

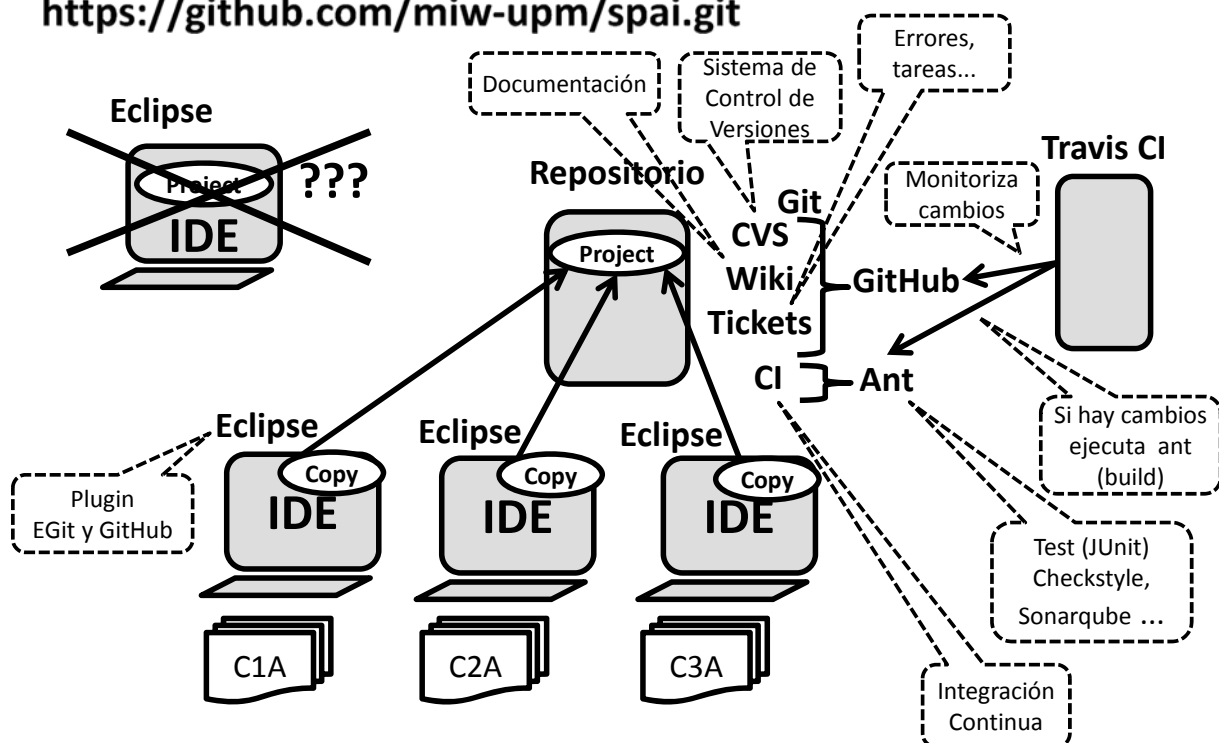
# Servicios y Protocolos de Aplicaciones en Internet

## SPAI (Entornos de desarrollo colaborativos)

Jesús Bernal Bermúdez


### Entorno de desarrollo colaborativo: Forja SPAI

<https://github.com/miw-upm/spai.git>



# Entorno de desarrollo colaborativo: Forja SPAI

<https://github.com/miw-upm/spai.git>

PUBLIC  miw-upm / spai Unwatch 1 Star 0 Fork 0

Forja: Eclipse (EGit + GitHub) + Travis CI + JUnit + Checkstyle — Edit

1 commit 1 branch 0 releases 1 contributor

branch: master spai

File	Commit	Time
miw-upm authored an hour ago	latest commit 22b46fadee	
.settings	D20H12:52	an hour ago
build	D20H12:52	an hour ago
lib	D20H12:52	an hour ago
src	D20H12:52	an hour ago
test	D20H12:52	an hour ago
.classpath	D20H12:52	an hour ago
.gitignore	D20H12:52	an hour ago
.project	D20H12:52	an hour ago
.travis.yml	D20H12:52	an hour ago
MIW_checkstyle.xml	D20H12:52	an hour ago
build.xml	D20H12:52	an hour ago

We recommend adding a README to this repository to help give people an overview of your project. [Add a README](#)

Code

Issues 3

Pull Requests 0

Wiki

Pulse

Graphs

Network

Settings

HTTPS clone URL

<https://github.com/miw-upm/spai.git>

You can clone with HTTPS, SSH, or Subversion

MiW

SPAI

15/10/2013

3

# Entorno de desarrollo colaborativo: Forja SPAI

<https://travis-ci.org/miw-upm/spai>

Ma Repositorios Recientes

miw-upm/spai 1  
Duración: 17 sec, Finalizado: 19 minutes ago

miw-upm/Erep 26  
Duración: 20 sec, Finalizado: about 20 hours ago

Forja: Eclipse (EGit + GitHub) + Travis CI + JUnit + Checkstyle

Actual Histórico Pull Requests Ramas

Build 1 Commit 22b46fadee (master)

State Passed Comparar a1014951b63\_22b46fadee47

Finalizado 19 minutes ago Autor miw-upm

Duración 17 sec Committer miw-upm

Mensaje D20H12:52

Configuración -

```
Using worker: worker-linux-2-2.bh.travis-ci.org:travis-linux-15
2
3 $ git clone --depth=50 --branch=master git://github.com/miw-upm/spai.git miw-upm/spai
4 $ cd miw-upm/spai
5 $ git checkout -qf 22b46fadee7d6a3c6a1e5f98c70797046357e
6 $ java -version
7 java version "1.7.0_21"
8 OpenJDK Runtime Environment (IcedTea 2.3.9) (7u21-2.3.9-buuntu0.12.04.1)
9 OpenJDK 64-Bit Server VM (build 22.7-b01, mixed mode)
10 $ javac -version
11 javac 1.7.0_21
12 $ ant test
13 Buildfile: /home/travis/build/miw-upm/spai/build.xml
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

MiW

SPAI

15/10/2013

4

# Entorno de desarrollo colaborativo: Forja

- ⊙ Consiste en un conjunto de aplicaciones para la administración y gestión de proyectos, con un interface unificado
  - Sistemas de control de versiones / repositorio de código
  - Listas de correo, foros y wiki
  - Servicios de hosting y descarga de ficheros
  - Sistemas de seguimiento de errores / tickets/ issues
- ⊙ Forjas Web
  - GitHub. <https://github.com/>. Sitio web que lleva integrada la forja. Para código abierto es gratuito, existe una versión de pago para proyectos privados
  - SourceForge. <http://sourceforge.net/>
  - Google Code. <https://code.google.com/intl/es/>
  - Otras: BerliOS, Gforce, Savannah...
- ⊙ Forjas en servidores propios
  - Sistema de control de versiones: Subversion, CVS, Git, Mercurial...
  - Sistema de tickets: Bugzilla, Trac, Redmine...
  - Wiki: Trac, Redmine, Wikimedia...
  - Listas de correo
  - Web: Apache + Drupal/Joomla...

## Sistema de control de versiones (SVC)

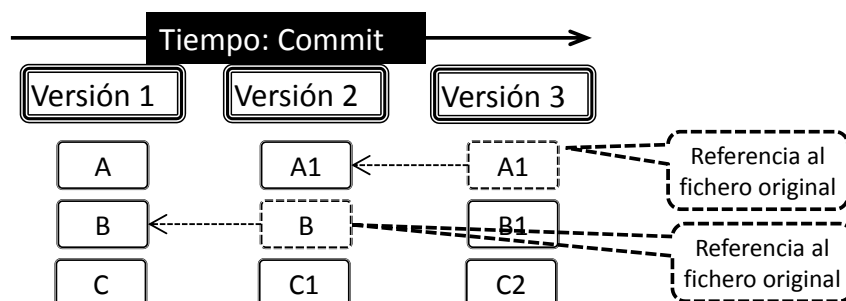
- ⊙ Es un sistema para el almacenamiento y gestión de los diversos cambios que se realizan sobre un conjunto de ficheros, pudiendo recuperar cualquier versión
- ⊙ En informática, se utiliza para controlar las versiones del código fuente: *repositorio de código*
- ⊙ Funcionalidades: Almacenamiento de código fuente, imágenes, documentos. Modificación y guardado de históricos
- ⊙ Tipos
  - Centralizados. Dependiente de un responsable. Facilita la gestión pero reduce la potencia y flexibilidad. Ejemplos: CVS, Subversion...
  - Distribuidos. Aumenta la flexibilidad pero complica la sincronización y gestión. Ejemplos: Git, Mercurial... En la actualidad se están imponiendo estos sistemas
- ⊙ Características típicas
  - La rama principal se conoce como *master*
  - Ramificaciones. Permite crear ramas auxiliares, normalmente se crea la rama *development*, para el desarrollo y la rama *release-\*.\**, para las versiones intermedias estables, rama *issues#n* para la corrección de bugs y ramas de subgrupos, para evitar colisiones...
  - Etiquetado (*Tag*). Permite poner etiquetado para identificar las versiones, una beta o el final de una iteración de desarrollo

# Git. Software de control de versiones distribuido

- ◉ Nace en 2005, tomando como experiencia el proyecto Bitkeeper (propietario)
- ◉ En 2008 nace GitHub, Forja en Web con repositorio basado en Git
- ◉ Documentación: <http://git-scm.com/book>
- ◉ Características generales
  - Auditoría del código: saber quién ha tocado qué y cuándo
  - Control sobre cómo ha cambiado nuestro proyecto con el paso del tiempo
  - Volver hacia atrás de una forma rápida
  - Control de versiones a través de etiquetas (*Tags*): versión 1.0, versión 1.0.1, versión 1.1, etc. Sabremos exactamente que había en cada una de ellas y las diferencias entre cualquiera de ellas dos
  - Seguridad: todas las estructuras internas de datos están firmadas con SHA1. No se puede cambiar el código sin que nos enteremos
  - Mejora nuestra capacidad de trabajar en equipo
  - Ramas (*branching*) y fusiones (*Merging*) muy eficientes

## Git. Fundamentos

- ◉ Control de versiones distribuido, cada cliente posee una copia completa del repositorio local de cada máquina utilizada
- ◉ Modelo de datos como instantáneas (*snapshot*) del sistema de archivos



- ◉ Trabaja sobre local, sin necesidad de conexión al remoto, muy rápido. Se podrá sincronizar con el remoto, pero con asistencia...
- ◉ Muy fiable, casi imposible perder el proyecto con tantas copias como participantes del proyecto

# Git. Fundamentos conceptuales

- ⊙ Estado de los archivos
  - Sin seguimiento (untracked). El fichero esta en el repositorio local
  - Preparado (staged). El fichero esta en el repositorio local y se encuentra preparado para almacenarse en el repositorio central en el siguiente *commit*
  - Modificado (modified). Se encuentra modificados en su repositorio local, pero no actualizado en el central
  - Confirmado (committed). Se encuentra actualizados en el repositorio central
- ⊙ Directorios
  - Git directory: repositorio, en formato comprimido (\*.git)
  - Working directory: área de trabajo. Es una copia descomprimida para que se puedan modificar
  - Staging area. Contiene un fichero con la información para actualizar en el siguiente commit

## GitHub. Instalación

- ⊙ Abrir una cuenta en GitHub: <https://github.com/>
  - Crear un repositorio. Activar la creación del fichero README.md
  - Borrar el repositorio: Settings / Delete this repository
- ⊙ Bajarse el cliente de Git: Set Up Git
  - <http://windows.github.com/>
- ⊙ Instalar y conectar con nuestro repositorio
- ⊙ Primeros pasos con GUI
  - GUI. Menu tools/open in explorer. Es una ventana estándar de ficheros, se pueden copiar ficheros directamente y modificarlos, se pueden actualizar los cambios mediante GUI Git, pulsando el botón de SHOW, en el marco de uncommitted chang
  - Una vez realizado el commit, habrá que realizar un *publish* o *sync* para actualizar el repositorio remoto

# GitHub. Instalación



MiW

SPAI

15/10/2013

11

# GitHub. Instalación



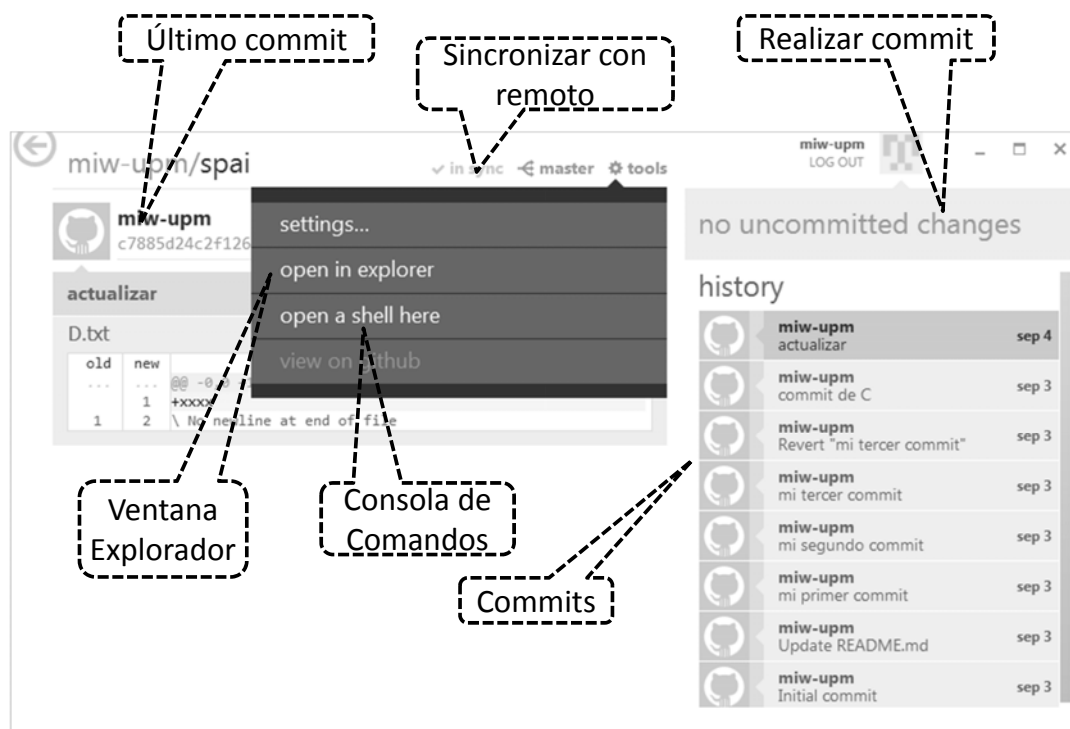
MiW

SPAI

15/10/2013

12

# GitHub. Instalación. Cliente Git



## GitHub. Primeros pasos con consola

- Consola. Menu tools/open a shell here. Se abre una ventana de comandos
  - `git help`. Ayuda de comandos
  - `git help [comando]`. Ayuda de un comando
  - `git log`. Para ver el histórico
  - `git config --list`. Para ver las variables de configuración
  - `git config user.name`. Para ver el nombre del usuario
  - `git config user.email`. Para ver el correo del usuario
  - `git config user.name [usuario]`. Para establecer el nombre del usuario
  - `git init`. Para iniciar un repositorio
  - `git clone [URL]`. Para clonar un repositorio
  - `git status`. Para ver el estado
  - `git add fich.ext` ó `"git add."` . Para añadir uno o todos los ficheros del directorio
  - `git commit -m "mi primer commit"`. Para confirmar los cambios en el repositorio
  - `git commit --amend`. Para modificar el último commit

# GitHub. Primeros pasos con consola

```
posh~git ~ git [master]
C:\Users\jbernal\Documents\GitHub\git [master] > git init
Reinitialized existing Git repository in C:\Users\jbernal\Documents\GitHub\git/.git/
C:\Users\jbernal\Documents\GitHub\git [master] > git help init
Launching default browser to display HTML ...
C:\Users\jbernal\Documents\GitHub\git [master] > git config user.name
miw-upm
C:\Users\jbernal\Documents\GitHub\git [master] > git config user.email
miw@eu1.upm.es
C:\Users\jbernal\Documents\GitHub\git [master] > git status
# On branch master
# nothing to commit, working directory clean
C:\Users\jbernal\Documents\GitHub\git [master] > ls
C:\Users\jbernal\Documents\GitHub\git [master] > ls

Directorio: C:\Users\jbernal\Documents\GitHub\git

Mode                LastWriteTime         Length Name
----                -
-a-----          06/10/2013   22:54             1 A.txt

C:\Users\jbernal\Documents\GitHub\git [master +1 ~0 -0 !] > git status
# On branch master
# Untracked files:
#   (use "git add <File>..." to include in what will be committed)
#
#       A.txt
nothing added to commit but untracked files present (use "git add" to track)
C:\Users\jbernal\Documents\GitHub\git [master +1 ~0 -0 !] > git add A.txt
C:\Users\jbernal\Documents\GitHub\git [master +1 ~0 -0] > git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   A.txt
C:\Users\jbernal\Documents\GitHub\git [master +1 ~0 -0] > git commit -m "Tercero"
[master b87ea09] Tercero
1 file changed, 1 insertion(+)
create mode 100644 A.txt
C:\Users\jbernal\Documents\GitHub\git [master] > git status
# On branch master
# nothing to commit, working directory clean
C:\Users\jbernal\Documents\GitHub\git [master] > git log
commit b87ea095147ccaa483ef3faaf3081567a9e68884
Author: miw-upm <miw@eu1.upm.es>
Date:   Sun Oct 6 22:55:21 2013 +0200

    Tercero

commit af85bc7afda6c7c95495ea227602fca4a9d87f
Author: miw-upm <miw@eu1.upm.es>
Date:   Sun Oct 6 22:52:55 2013 +0200

    Segundo

commit e478cb7e6212463412a48847a37276a815a90de8
Author: miw-upm <miw@eu1.upm.es>
Date:   Sun Oct 6 22:50:38 2013 +0200

    Inicial
C:\Users\jbernal\Documents\GitHub\git [master] > git push
```

Se consulta el estado

Se crea el fichero A.txt

Se prepara A.txt para el próximo commit

Se realiza commit

Se consulta los commit

Se actualiza el repositorio remoto

MiW

SPAI

15/10/2013

15



## Mi primer Git

- Crear un repositorio llamado "git1"
- Realizar la instalación del cliente y crear varios ficheros desde GUI
- Realizar varios commit
- Crear un fichero consola.txt y realizar todo el proceso de commit desde consola

MiW

SPAI

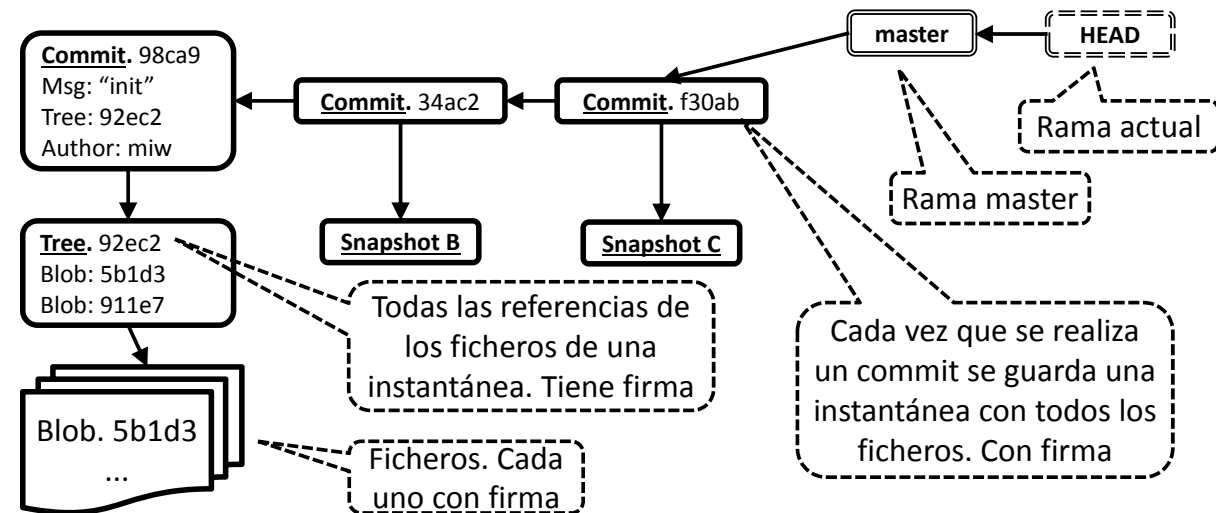
15/10/2013

16



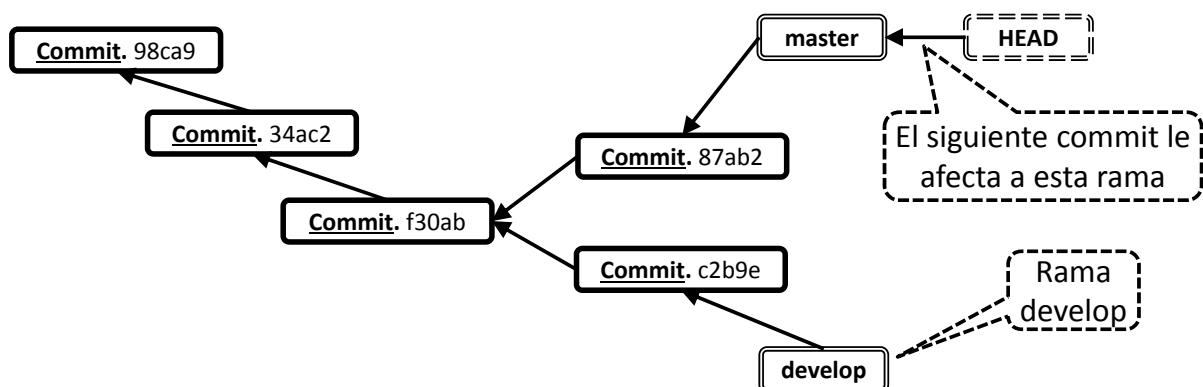
# GitHub. Instantánea de ficheros. Ramas

- Cada commit guarda una instantánea de todos los ficheros
- Cada commit tiene una referencia al anterior commit
- Una rama es una referencia a un commit
- *HEAD*. Es una referencia de la rama actual



# GitHub. Instantánea de ficheros. Ramas

- Comandos
  - `git branch develop`. Crea una rama llamada "develop"
  - `git checkout develop`. Se indica que la rama actual es "develop", es decir, "HEAD" apunta a "develop"
  - `git commit -m "se aplica a la rama actual"`
  - `git merge rama`. Une la rama actual con la indicada



# GitHub y Eclipse: EGit

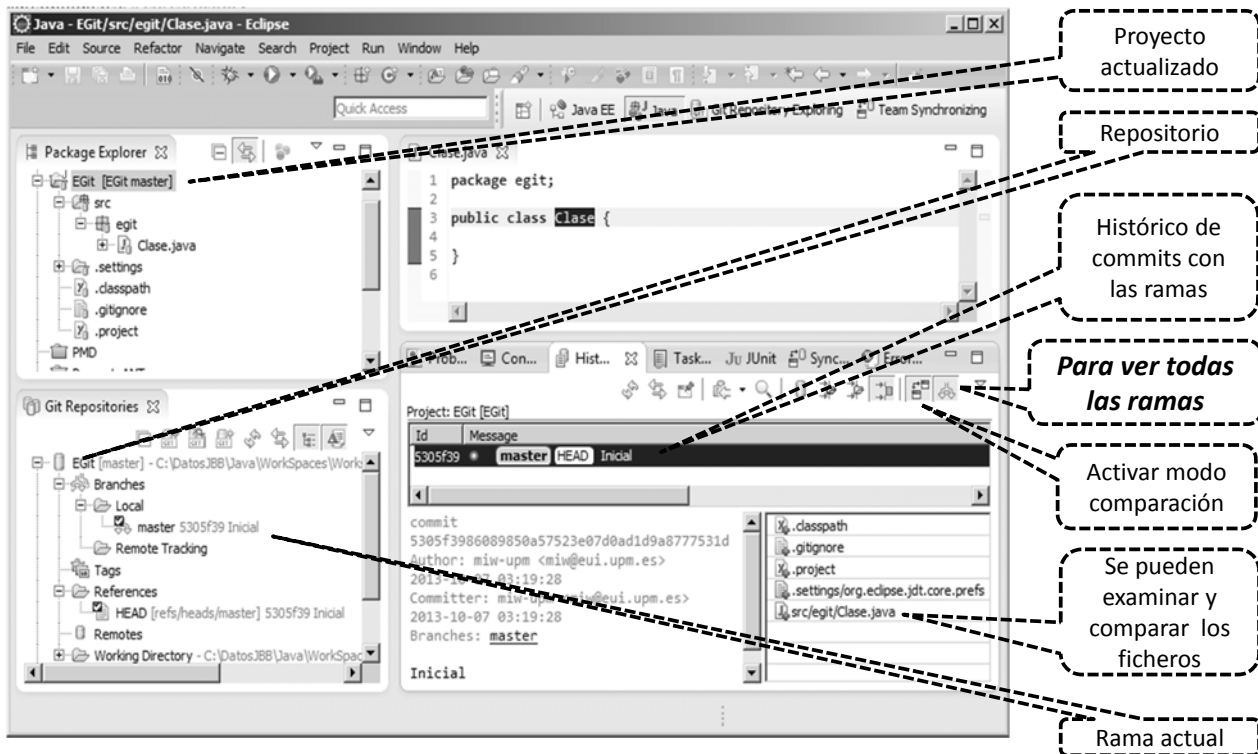
- ⦿ El Plugin de Eclipse para conectar con Git es EGit
- ⦿ A partir de la versión de Eclipse Kepler (sep 2013), viene instalado, las anteriores se debe instalar
- ⦿ Documentación: [http://wiki.eclipse.org/EGit/User\\_Guide](http://wiki.eclipse.org/EGit/User_Guide)



## EGit

1. Se crea un proyecto de tipo *Java Project*.
2. Compartir el proyecto
  - Menú contextual del proyecto >>**Team>Share Project**, elegir **Git** y pulsar el botón de **Create...**, marcar el checkbox de **Use or create repository in parent folder of project**
3. Elegir la ruta del workspace del proyecto, crear el repositorio y finalizar
4. Visualizar la ventana de repositorios
  - Menú contextual del proyecto >>**Team>Show in Repositories View**
5. Primer commit
  - Crear un paquete java "egit" y una clase "Clase"
  - Menú contextual del proyecto >>**Team>Commit...**, rellenar el mensaje y marcar los ficheros a subir. Si se comparte el proyecto entre equipos, se recomienda marcar todos los ficheros
  - Pulsar el botón de **Commit**
6. Visualizar la ventana del histórico
  - Menú contextual del proyecto >>**Team>Show in History**

# EGit



MiW

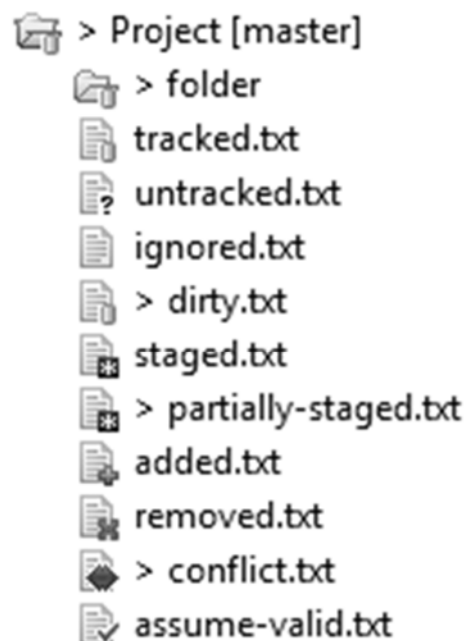
SPAI

15/10/2013

21

## EGit. Decoraciones de ficheros

- ◉ Tracked. Actualizado en el repositorio
- ◉ Untraked. Sin seguimiento
- ◉ Ignored. El fichero no se guarda en el repositorio
- ◉ Dirty. Modificado y preparado para guardarse en el repositorio en el siguiente commit
- ◉ Staged. Tiene cambios que se han añadido al índice
- ◉ Partially-staged. Tiene cambios en el índice y en la zona de trabajo
- ◉ Added. Se ha añadido al repositorio
- ◉ Removed. ha eliminado del repositorio
- ◉ Conflict. Existe conflicto en el fichero que no se puede resolver automáticamente, se debe intervenir manualmente
- ◉ Assume-valid. Asumir sin cambios



MiW

SPAI

15/10/2013

22

# EGit. Acciones

- Crear rama y hacerla activa
  - En la ventana de *Git Repositories*, en el apartado Branches, enfocar Local y en el menú contextual: >>**Switch to>New Branch...**
- Activar una rama
  - Enfocar la rama, en el menú contextual: >>**Checkout**
- Fusionar la rama *develop* en la rama *master*
  1. Activar la rama *master*
  2. Enfocar la rama *master*, menú contextual, >>**Merge... ó Rebase**
    - Sin conflictos, se hace automáticamente
    - Con conflictos, se debe hacer manualmente
      1. Editar los ficheros con conflictos y modificar sus fuentes, guardar
      2. Se enfoca, menú contextual, >>**team>Add to Index**. El icono cambia a estado *staged*
      3. Se realiza un commit
  1. Se puede borrar la rama *develop*, menú contextual de la rama a borrar: >>**Delete Branch**
- Crear un Tag
  - En el apartado Tags, menú contextual, >>**Create Tags...**

MiW

SPAI

15/10/2013

23

# EGit. Acciones

Se pueden comparar dos versiones del documento

1: 3 commit en la rama master

2: se crea la rama develop

3: 2 commit en la rama develop

4: 2 commit en la rama master

5: Se fusiona las ramas y se crea un Tag

6: 2 commit en la rama master

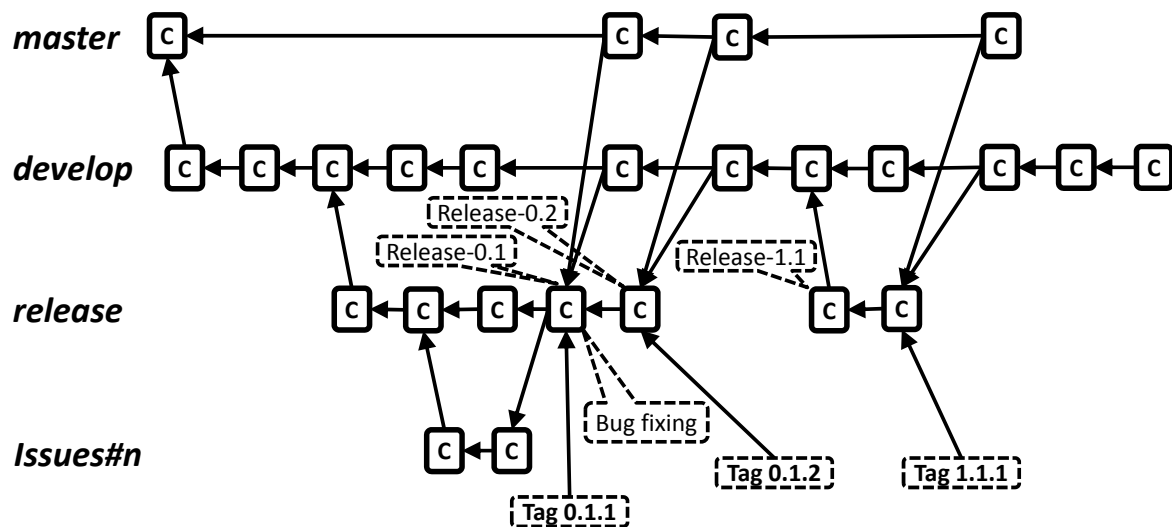
MiW

15/10/2013

24

# Git. Ramas

## Desarrollo de software



## GitHub. Repositorios remotos

- ◉ La sincronización con el repositorio remoto debe realizarse manualmente
- ◉ Operaciones básicas
  - Actualizar el repositorio remoto, desde el repositorio local: **push**. Si alguna rama del repositorio remoto ha sido modificada por un tercero, la actualización de dicha rama será rechazada; primero deberemos actualizar el repositorio local con los últimos cambios
  - Actualizar el repositorio local, desde el repositorio remoto: **pull** o **fetch**
    - **fetch**. Se recuperan los cambios remotos, pero deberemos inspeccionar y fusionar manualmente
    - **pull**. Se recuperan los cambios remotos y se fusionan con el repositorio local
- ◉ Comandos
  - **git clone [URL]**. Para clonar un repositorio
  - **git remote -v**. Para ver los repositorios remotos
  - **git remote show [repo]**. Para ver los datos de un repositorio
  - **git fetch [repo]**. Para actualizar el repositorio local
  - **git pull [repo]**. Para actualizar el repositorio local y fusionar
  - **git push [repo] [rama]**. Para actualizar el repositorio remoto por completo o solo una rama

# EGit. Repositorios remotos

## Actualización del remoto

1. Se genera una llave privada
    - Menu >>**Window>Preferences>General>Network Connections>SSH2**, en la ficha **Key Management**, se genera una clave **DSA**, se guarda en disco y se debe copiar para registrarla en GitHub
    - GitHub, en el menú **Account settings**, en la ficha de **SSH Keys**, se da de alta la clave
  2. Asociar un repositorio remoto
    - Ir a la web de GitHub y copiar la URL de SSH del repositorio
    - En la ventana de **Git repositories**, en **Remotes**, en el menú contextual **Create Remote...**
    - Se puede dejar el nombre **origin**
    - Pulsar el botón **Change...**, rellenar la URL y elegir el protocolo SSH
    - En el cuadro de **Ref mappings**, se configuran las ramas a actualizar. Pulsar el botón de **Advanced...**, se elige la referencia (**refs/heads/master** ó **refs/heads/develop**) o, en nuestro caso, pulsar el botón de **Add All Branches Spec**, para actualizar todas las ramas (**refs/heads/\***)
    - Pulsar el botón de **Save**
  3. En el menú contextual del repositorio remoto, realizar un **Push**. Comprobar en la web de GitHub que se ha subido el proyecto
- ☉ Cada vez que queramos actualizar el repositorio remoto, desde el repositorio local, se deberá realizar un **Push**



## EGit1. Manejo de ramas desde EGit

### Realizar la siguiente secuencia

1. Crear la llave **SSH2** en Eclipse y **registrarla** en GitHub
2. Crear un proyecto Java nuevo (**egit**) Rama master: commit inicial. Marcar todos los ficheros de Eclipse
3. Crear un repositorio nuevo en la Web GitHub: **EGit1**. Crearlo sin ficheros
4. En Eclipse, crear un nuevo repositorio remoto con la URL de EGit1
5. Crear rama **develop**: realizar tres cambios en la clase y 3 commit
6. Crear la rama **release-1.0**: realizar 1 cambio y un 1 commit
7. Rama. develop: realizar 2 commit
8. Rama. release-1.0: 2 commit. Se supone que esta versión es estable
9. Realizar **Push**
10. Crear un **Tag** a la versión estable
11. Fundir la rama **release con master** y **release con develop**
12. Realizar **Push**
13. Rama. develop: 2 commit
14. Realizar **Push**

# EGit. Repositorios remotos

## Actualización del local

- ◉ Si alguna rama del remoto ha cambiado por un tercero, antes de hacer **push**, se debe integrar los cambios en nuestro repositorio local
- ◉ Ir a la Web de GitHub y cambiar el repositorio
- ◉ Configurar el **fetch**, para que actualice todas las ramas (**refs/remotes/origin/\***)
- ◉ Realizar **Fetch**. Los cambios del remoto aparecen como un nuevo commit con referencia **Origin**
- ◉ Activar la rama que queramos que se fusionen los cambios (**checkout**)
- ◉ Enfocar la referencia del remoto, menú contextual, **Merge**
- ◉ Por último, ya podemos hacer **Push**

# EGit. Repositorios remotos

## Actualización del local

The screenshot shows the EGit interface with a commit history on the left and a 'Push Results' dialog in the center. The commit history lists commits b5f7770 (master HEAD), d6a7457 (CM1), 381770e (develop CD2), 703445d (CD1), and 1ff6598 (Inicial). The 'Push Results' dialog shows a rejected push for the 'develop' branch because it is not a fast-forward. A 'Message Details' section shows the repository URL: ssh://git@github.com:miw-upm/EGit.git. Four numbered callouts explain the steps: 1. A commit is made and the repository is updated. 2. A file is created in the 'develop' branch from the GitHub web. 3. A modification is made in the 'develop' branch and a commit and push are performed. 4. The push is rejected because a modification was detected.

1: Se realiza un **commit** y se actualiza el repositorio.

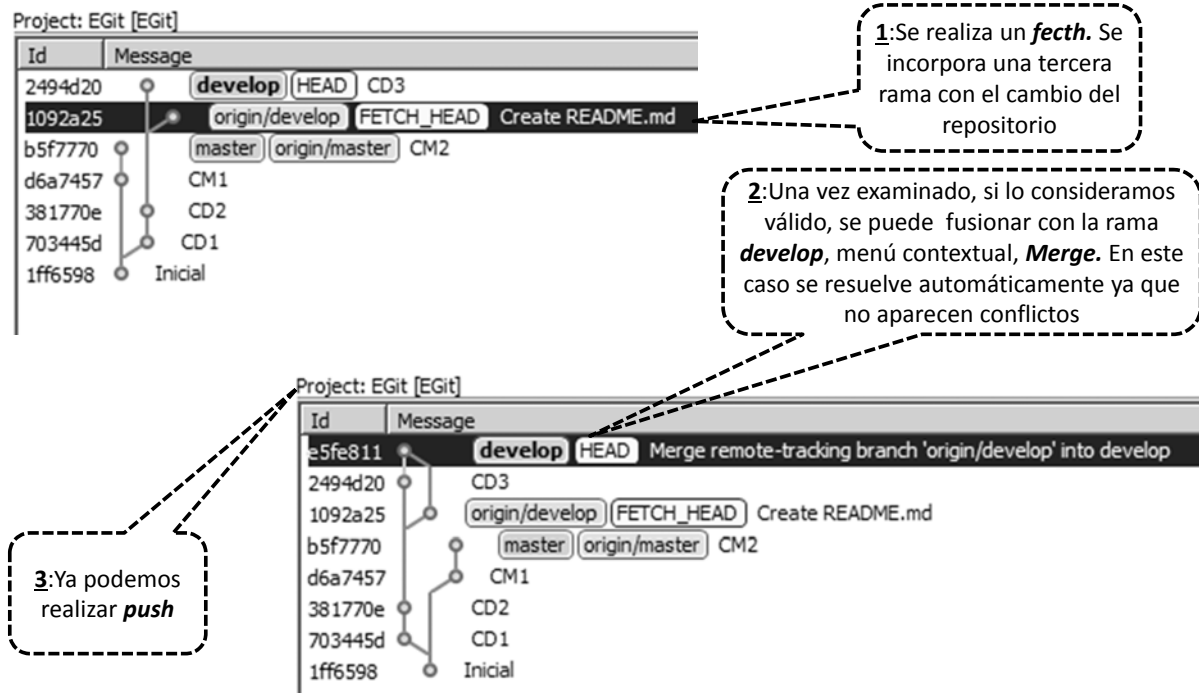
2: En la rama "develop", se crea un fichero desde la web de GitHub

3: Se realiza una modificación en la rama "develop" y se realiza **commit** y **push**

4: Se rechaza el **push** en la rama "develop", porque se detecta que se ha modificado

# EGit. Repositorios remotos

## Actualización del local



MiW

SPAI

15/10/2013

31

## EGit. Importar proyecto desde el repositorio

1. Establecer la perspectiva de *Git Repository Exploring*
2. Pulsar el botón de **Clone a Git Repository**
  - Copiar la URL *https* de GitHub del repositorio a clonar y establecer el protocolo *https*, y el usuario y contraseña
  - Establecer la ruta del repositorio local
3. Menú >>**Archivo>import>Git>Projects from Git**
  - Pulsar Local
  - Buscar el proyecto en la ruta local
4. Cambiar a la perspectiva de Java y debe aparecer el proyecto
5. Si queremos realizar **Push** sobre el repositorio, se deberá crear una llave SSH

MiW

SPAI

15/10/2013

32





## EGit1B

### Compartir repositorios remotos con EGit

- ⦿ Al ejercicio EG1 anterior, añadir la siguiente secuencia
  1. Desde la web de GitHub, en la rama **develop**, crear un fichero
  2. Desde Eclipse, en la rama **develop**, realizar un cambio y **commit**
  3. Intentar un **Push**, será rechazado
  4. Integrar los cambios en nuestro repositorio local, para poder finalmente realizar ese **Push**

## GitHub. Tickets

- ⦿ Milestone: Hito (título, descripción y fecha)
- ⦿ Issue: tarea, error, ticket... (título, asignado a, asociado a un hito, comentario, etiquetas) (abierto-cerrado)
  - Se abre un foro de comentarios
- ⦿ Labels: etiquetas, pueden crearse etiquetas propias

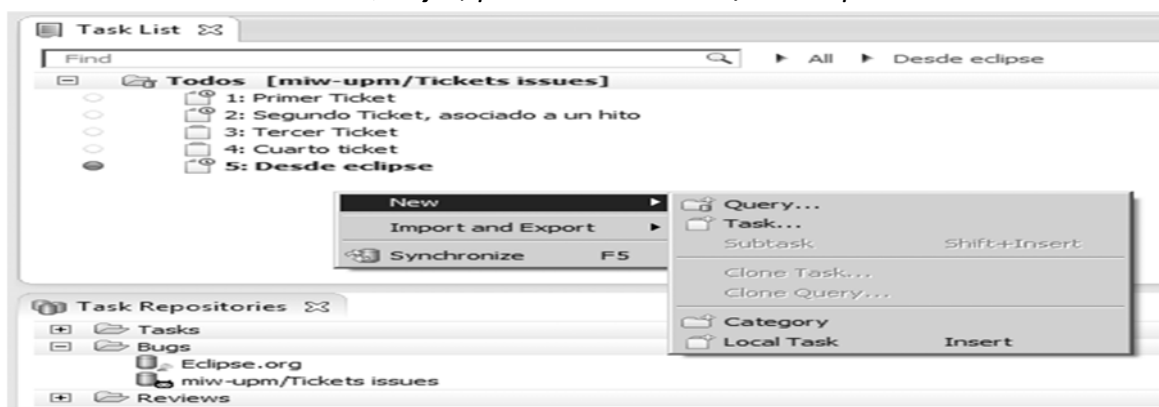
## IDE: Eclipse. GitHub. Tickets

- ◉ El Plugin de Eclipse para conectar con GitHub es **GitHub Mylyn Connector**
- ◉ Se debe instalar, existen dos procedimientos:
  - Install New Software... <http://download.eclipse.org/egit/github/updates-nightly>
  - Eclipse Marketplace...
- ◉ Documentación: <http://wiki.eclipse.org/EGit/GitHub/UserGuide>



## GitHub. Tickets

- ◉ Importar el repositorio. Menú **>>File>Import...>Tasks>GitHub Task Repositories**, identificar con usuario-contraseña, y elegir el repositorio
- ◉ Mostrar el repositorio de tareas. Menú **>>Window>Show View>Mylyn>Task Repositories**
- ◉ Mostrar la ventana de lista de tareas. Menú **>>Window>Show View>Mylyn>Task List**
- ◉ Se deben realizar una **Query...**, para leer las tareas/tickets pendientes



# GitHub. Wiki

### [Lenguaje Markdown] (<http://daringfireball.net/projects/markdown/>)

Línea en blanco para separar estructuras

"\" carácter de escape para los caracteres especiales de Markdown

Con 4 blancos al principio: \<pre> <code>

Salto de línea: 2 blancos al final

##### Cabecera 6

### Cabecera 3

*\*Italica\**   **\*\*Negrita\*\***   ***\*\*\*ItalicaNegrita\*\*\****  
*\_Italica\_*   \_\_Negrita\_\_   \_\_\_ItalicaNegrita\_\_\_

1. lista A

1. lista B

\* lista B1

\* lista B2

1. lista C

> Sangría

>> Sangría doble

>> Sangría doble. Para finalizar, línea en blanco

> Sangría

Se debe dejar una línea en blanco para introducir HTML

```
<table><tr><td>Celda de tabla</td><td>otra</td></tr></table>
```

```
```java
```

```
public class Suma {  
    public double getSuma(double operando1, double operando2) {  
        return operando1 + operando2;  
    }  
}
```

<http://www.google.com/> Una URL se convierte en un enlace

[Google](<http://www.google.com>)

[Google][ ]

[Google]: <http://www.google.com>

[Volver](../wiki)



SPAI

15/10/2013

37

# GitHub. Wiki

## Lenguaje Markdown

Con 4 blancos al principio: \<pre> <code>

### Cabecera 6

*Italica* **Negrita** *ItalicaNegrita*

*Italica* **Negrita** *ItalicaNegrita*

1. lista A

◦ lista B1

2. lista C

Sangría

Sangría doble

Sangría doble. Para finalizar, línea en blanco

Sangría

Celda de tabla	otra
----------------	------

```
public class Suma {  
    public double getSuma(double operando1, double operando2) {  
    }  
}
```

<http://www.google.com/> Google

Volver



SPAI

15/10/2013

38

- ⦿ <https://github.com/miw-upm/EGit2>
  - Leer el fichero README.md
- ⦿ <https://github.com/miw-upm/EGit2/wiki>

## Ant. Construcción de proyectos

- ⦿ <http://ant.apache.org/>
- ⦿ Es una herramienta de construcción de proyectos (build) desarrollada por la *Apache Software Foundation*
- ⦿ Realiza las diferentes etapas como: compilación, generar los docs, pasar los test, generar el jar...
- ⦿ Se apoya en un fichero de configuración: *build.xml*, para establecer las diferentes etapas
- ⦿ Es la más fácil, pero tiene carencias, debe utilizarse XML y no gestiona las dependencias
- ⦿ Esta Integrada con Eclipse
  - Crear build.xml en la raíz del proyecto
  - Registrar. Menu: >>**Window>General>Editors>File\_Associations**
  - Enfocar build.xml... Run as..
- ⦿ Alternativas:
  - Maven (<http://maven.apache.org/>), es la más usada en la actualidad, pero más complicada. Sigue utilizando XML y gestiona las dependencias
  - Gradle (<http://www.gradle.org/>). Viene empujando fuerte. Posee las características de Ant y Maven, pero se apoya en el lenguaje Groovy, un lenguaje adaptado a tareas y más fácil de mantener. Si se sabe Ant, la migración es sencilla

## Ant. Construcción de proyectos. Fuentes

```
public class Suma {
    public double getSuma(double operando1, double operando2) {
        return operando1 + operando2;
    }
    public double incrementa(double operando) {
        return operando + 1;
    }
}

public class SumaTest {
    private Suma suma;
    @Before public void paraEjecutarAntes() {
        suma = new Suma();
    }
    @Test public void aVerSiIncrementaBien() {
        assertEquals("Test incrementa", 2.0, suma.incrementa(1.0), 1e-6);
    }
    @Test public void aVerSiSumaBien() {
        assertEquals("Test suma", 2.0, suma.getSuma(1.0, 1.0), 1e-6);
    }
    @After public void paraEjecutarDespues() {
        // Fin de test. Aquí liberar recursos o borrar rastros del test
    }
}
```

## Ant. Construcción de proyectos. build.xml(1-2)

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="ProyectoANT" default="main" basedir=".">

    <!-- Variables globales -->
    <property name="src" location="src" />
    <property name="test" location="test" />
    <property name="build" location="build" />
    <property name="dist" location="dist" />
    <property name="junit" location="lib/junit-4.10.jar" />
    <property name="testReport" location="testReport" />
    <property name="name" value="ProyectoANT" />
    <property name="version" value="1.0.0" />

    <!-- Se resuelven las dependencias -->
    <path id="classpath.test">
        <pathelement location="${build}/main" />
        <pathelement location="${build}/test" />
        <pathelement location="${junit}" />
    </path>

    <!-- Borrando directorios... -->
    <target name="clean">
        <delete dir="${build}" />
        <delete dir="${dist}" />
        <delete dir="${testReport}" />
    </target>

    <!-- Creando directorios... -->
    <target name="init">
        <mkdir dir="${build}/main" />
        <mkdir dir="${build}/test" />
        <mkdir dir="${dist}" />
        <mkdir dir="${testReport}" />
    </target>

</project>
```

## Ant. Construcción de proyectos. build.xml(2-2)

```
<!-- Compilando... -->
<target name="compila" description="Compila">
  <javac includeantruntime="false" srcdir="${src}" destdir="${build}/main" classpathref="classpath.test" />
  <javac includeantruntime="false" srcdir="${test}" destdir="${build}/test" classpathref="classpath.test" />
</target>

<!-- Ejecutar test JUnit, salida en fichero XML -->
<target name="test" depends="compila">
  <junit printsummary="on" fork="yes" haltonfailure="true">
    <classpath refid="classpath.test" />
    <batchtest fork="yes" todir="${testReport}">
      <formatter type="xml" usefile="true" />
      <fileset dir="${test}">
        <include name="**/SumaTest.java" />
        <!-- <include name="**/*Test*.java" /> -->
      </fileset>
    </batchtest>
  </junit>
</target>

<target name="empaqueta" depends="compila, test" description="Empaqueta">
  <jar jarfile="${dist}/lib/${name}v${version}.jar" basedir="${build}">
    <manifest>
      <attribute name="Built-By" value="${user.name}" />
      <attribute name="Specification-Title" value="${name}" />
      <attribute name="Specification-Version" value="${version}" />
      <attribute name="Main-Class" value="es.upm.miw.spai.ant.test.SumaTest" />
    </manifest>
  </jar>
</target>

<!-- Principal... -->
<target name="main" depends="clean, init, compila, test, empaqueta" description="Compila y empaqueta">
  <echo message="Compilando y creando el archivo .jar ..." />
</target>

</project>
```

## Ant. Construcción de proyectos. Eclipse



# Integración Continua (CI)

- ⊙ La integración continua es una práctica de desarrollo de software, propuesto inicialmente por Martin Fowler, la cuál los miembros de un equipo integran su trabajo frecuentemente (diariamente) y se revisa automáticamente
- ⊙ Filosofía muy relacionada con las metodologías ágiles y la programación extrema (XP)
- ⊙ Cada integración se verifica mediante una herramienta de construcción para detectar los errores de integración tan pronto como sea posible
- ⊙ Para ello se pueden utilizar un software especializado en automatizar tareas y que estas se ejecuten de forma automática. Las tareas a realizar pueden ser:
  - Compilación de los componentes
  - Obtener métricas de calidad de código
  - Ejecución de pruebas unitarias
  - Ejecución de pruebas de integración, de aceptación
- ⊙ Buenos principios
  - Contribuir a menudo
  - No contribuir con código roto
  - Soluciona los builds rotos inmediatamente
  - Escribe tests automáticos
  - Todos los tests deben pasar

## Integración Continua (CI). Procedimiento

- ⊙ El equipo modifica el código, y una vez testeado, se realiza un ***commit*** + ***push*** al repositorio
- ⊙ La herramienta de CI monitoriza el repositorio y se dispara cada vez que detecta un cambio
- ⊙ CI ejecuta la construcción del todo el código fuente, aplicando los testeos de control de calidad, pruebas unitarias, pruebas de integración...
- ⊙ CI envía correos de los resultados de la integración del nuevo código

# Integración Continua. Herramientas

- ◎ Travis CI
  - <https://travis-ci.org/>
  - Documentación: <http://about.travis-ci.org/docs/>
  - Travis CI es un sistema de integración continua gratuito en la nube para la comunidad OpenSource. Está integrado con GitHub y ofrece soporte para: Java, Groovy, Haskell, Node.js, PHP, Python, Ruby...
- ◎ Jenkins
  - <http://jenkins-ci.org/>
  - Jenkins es un software de integración continua de código abierto escrito en Java, evolución de Hudson
  - Jenkins tiene soporte para sistemas de control de versiones, algunas como SVN, CVS, Git y corre en un servidor de aplicaciones como por ejemplo Tomcat o Jboss permitiendo la ejecución de proyectos Ant, Maven...

## CI. Herramientas de análisis estáticos

### Checkstyle

- ◎ <http://checkstyle.sourceforge.net/>
- ◎ *Checkstyle* es una herramienta de desarrollo que ayuda a los programadores a escribir código Java adecuándose a estándares de codificación establecidos, facilitando para ello la automatización del proceso
- ◎ *Checkstyle* incorpora las recomendaciones de *Oracle* sobre el estilo del código, pero las reglas pueden ser redefinidas y ampliadas
- ◎ *Checkstyle* se integra en varios IDEs a través de distintos plugins. Para Eclipse, se encuentra en la ruta:
  - <http://eclipse-cs.sf.net/update/>
- ◎ Ejemplos de reglas
  - **AbstractClassName.** Se asegura que los nombres de las clases abstractas se ajusten a una expresión regular
  - **BooleanExpressionComplexity.** Restringe los operadores booleanos anidados (&&, ||, &, | y ^) a una profundidad especificada (por defecto = 3)
  - **EmptyBlock.** Comprueba si hay bloques vacíos
  - **FileTabCharacter.** Comprueba si el archivo contiene caracteres de tabulación
  - **Indentation.** Comprueba la correcta sangría de código Java



# CI. Herramientas de análisis estáticos

## PMD y FindBugs

- ◉ PMD es un analizador estático de código que utiliza unos conjuntos de reglas para identificar problemas dentro del software
  - Posibles defectos: sentencias try/catch/finally/switch vacías
  - Código muerto: variables, parámetros y métodos no utilizados
  - Código no óptimo: uso ineficiente del StringBuffer, etc.
  - Expresiones innecesarias: sentencias "if" innecesarias
  - Código duplicado: el código copiado y pegado significa copiar y pegar defectos
  - <http://pmd.sourceforge.net/>
- ◉ FindBugs es un proyecto OpenSource de la universidad de Maryland. Este programa sirve para analizar código implementado bajo la especificación Java (bytecode) y encontrar posibles bugs
  - <http://findbugs.sourceforge.net/>

# CI. Herramientas de análisis estáticos

## SonarQube

- ◉ <http://www.sonarqube.org/>
- ◉ SonarQube (antes Sonar) es una plataforma de código abierto para el control continuo de la calidad del software.
- ◉ Permite analizar distintos parámetros de calidad, como la complejidad ciclométrica, el grado de replicación de código, la cobertura con tests o los estándares de codificación



# Eclipse, GitHub y Travis CI

1. Crear un proyecto Eclipse y conectarlo a un repositorio de GitHub, **se debe poner el repositorio local en el workspace de Eclipse**
  2. Para empezar a Travis CI, acceder a través de GitHub OAuth. Ir a Travis CI y seguir el vínculo de iniciar sesión, situado en la parte superior: *Sign in GitHub*
  3. Activar en GitHub el Servicio Hook#. Una vez que has iniciado sesión, en la página del perfil, activar el interruptor para cada repositorio que desea conectar en Travis CI
  4. Añadir archivo *“.travis.yml”* al raíz del proyecto. El contenido podrá ser:
    1. language: java
    2. script: ant test
- ⊙ A partir de ahora, el repositorio estará monitorizado por Travis CI. Cada vez que se realice un commit, ejecuta uno de los siguientes constructores en sus servidores, esta operación puede ir despacio (varios minutos):
- Maven: pom.xml y ejecuta *“mvn test”*
  - Gradle: build.gradle y ejecuta *“gradle check”*
  - Ant: build.xml y ejecuta *“ant test”*

## Travis CI. Ant

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="ProyectoANT" default="test" basedir=".">

  <!-- Variables globales -->
  <property name="src" location="src" />
  <property name="build" location="build" />
  <property name="dist" location="dist" />
  <property name="test" location="test" />
  <property name="junit" location="lib/junit-4.10.jar" />
  <property name="testReport" location="testReport" />
  <property name="name" value="ProyectoANT" />
  <property name="version" value="1.0.0" />

  <!-- Se resuelven las dependencias -->
  <path id="classpath.test">
    <pathelement location="${build}/main" />
    <pathelement location="${build}/test" />
    <pathelement location="${junit}" />
  </path>

  <!-- Creando directorios... -->
  <target name="init">
    <mkdir dir="${build}/main" />
    <mkdir dir="${build}/test" />
  </target>
</project>
```

# Travis Cl. Ant

```
<!-- compilar... -->
<target name="compila" description="Compila">
  <javac includeantruntime="false" srcdir="${src}" destdir="${build}/main"
    classpathref="classpath.test" />
  <javac includeantruntime="false" srcdir="${test}" destdir="${build}/test"
    classpathref="classpath.test" />
</target>

<!-- Principal... -->
<target name="test" depends="init, compila" description="Integracion Continua">
  <echo message="Integracion Continua..." />
  <junit printsummary="on" fork="yes" haltonfailure="true">
    <classpath refid="classpath.test" />
    <batchtest fork="yes" >
      <formatter type="brief" usefile="false" />
      <fileset dir="${test}">
        <include name="**/SumaTest.java" />
        <!-- <include name="**/*Test*.java" /> -->
      </fileset>
    </batchtest>
  </junit>
</target>

</project>
```

# Checkstyle. Ant. build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="checkstyle" default="main" basedir=".>
  <!-- Variables globales -->
  <property name="src" location="src" />
  <property name="checkstyle.lib" location="lib/checkstyle-5.6-all.jar" />
  <property name="checkstyle.checks" location="MIW_checkstyle.xml" />
  <property name="checkstyle.report" location="reports.xml" />
  <!-- Ejecutar checkstyle, salida en fichero XML -->
  <target name="checkstyle">
    <taskdef resource="checkstyletask.properties"
      classpath="${checkstyle.lib}" />
    <checkstyle config="${checkstyle.checks}"
      failureProperty="checkstyle.failure" failOnViolation="false">
      <formatter type="xml" tofile="${checkstyle.report}" />
      <fileset dir="${src}" includes="**/*.java" />
    </checkstyle>
  </target>
  <!-- Principal... -->
  <target name="main" depends="checkstyle" description="Checkstyle">
    <echo message="testando con checkstyle..." />
  </target>
</project>
```

# Checkstyle. Ant. build.xml. Travis CI

```
<!-- Variables globales -->
<property name="checkstyle.lib" location="lib/checkstyle-5.6-all.jar" />
<property name="checkstyle.checks" location="MIW_checkstyle.xml" />

<!-- Ejecutar checkstyle, salida por consola -->
<target name="checkstyle">
    <taskdef resource="checkstyletask.properties"
        classpath="${checkstyle.lib}" />
    <checkstyle config="${checkstyle.checks}"
        failureProperty="checkstyle.failure" failOnViolation="false">
        <formatter type="xml" usefile="false" />
        <fileset dir="${src}" includes="**/*.java" />
    </checkstyle>
</target>
```



## SPAI.EGit2B

Trabajo colaborativo, con TRAVIS-CI y ANT

1. Una vez completado el ejercicio EGit2B, integrarlo con TRAVIS-CI y ANT
2. Realizar una modificación y comprobar la CI



# Solitario

## Trabajo colaborativo, Solitario

1. <https://github.com/miw-upm/solitario/wiki>