

# FOSS4G-fr 2016

## QGIS Server Plugins et API Python

# 3Liz SARL

Création en Mars 2007

QGIS / LizMap / QGIS Server

Cadastre / QuickOSM / LayerBoard



11/05/2016

FOSS4G-fr 2016



# FOSS4G-fr 2016

QGIS Server  
Plugins et API Python

# QGIS Server



# Origines de QGIS Server

- Lancé en 2006 :
  - Au sein de projets de recherche
    - 'Orchestra' (Infrastructure de données spatiales européennes pour la gestion des risques)
    - 'SANY' (Sensor Anywhere)
  - Institute of Cartography (ETH Zurich)
  - Marco Hugentobler (SourcePole)

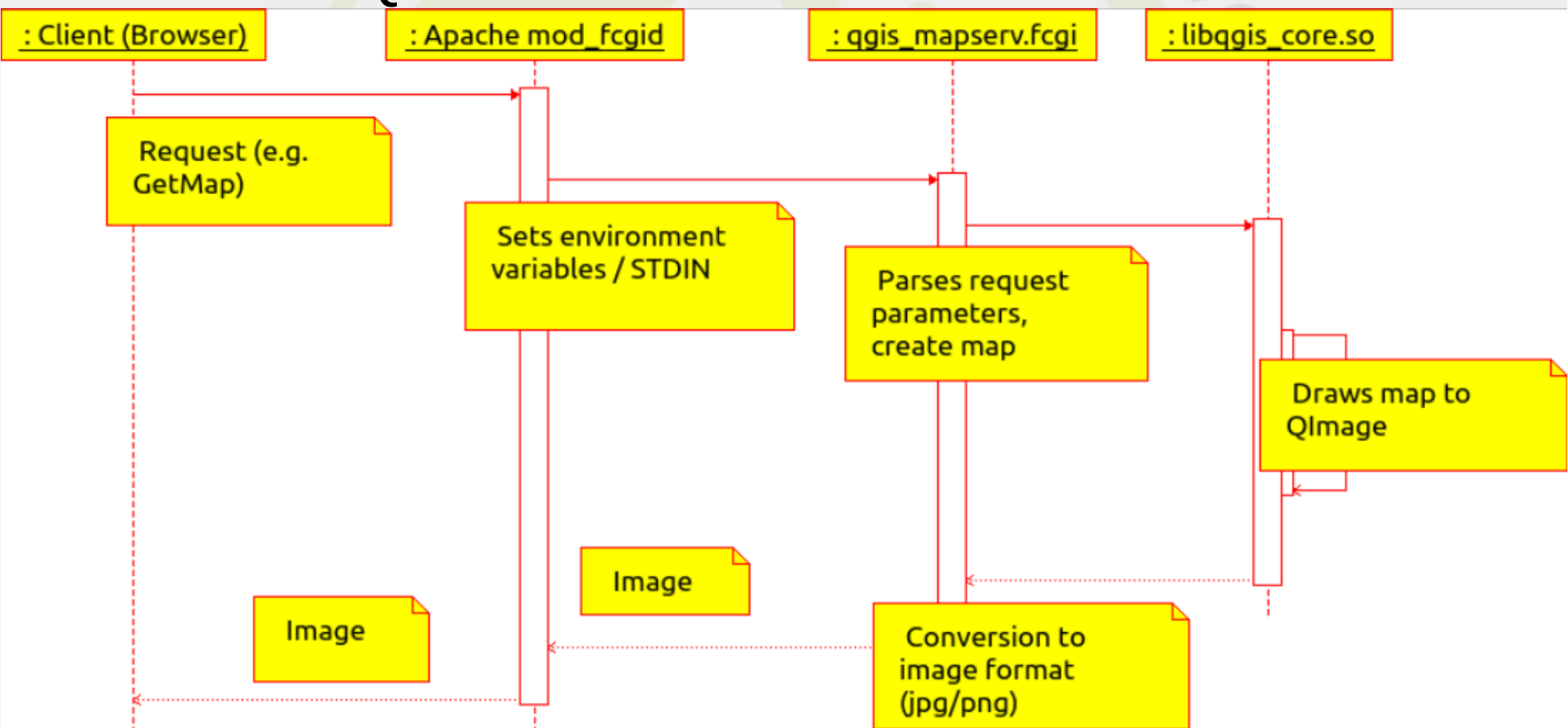


# Origines de QGIS Server

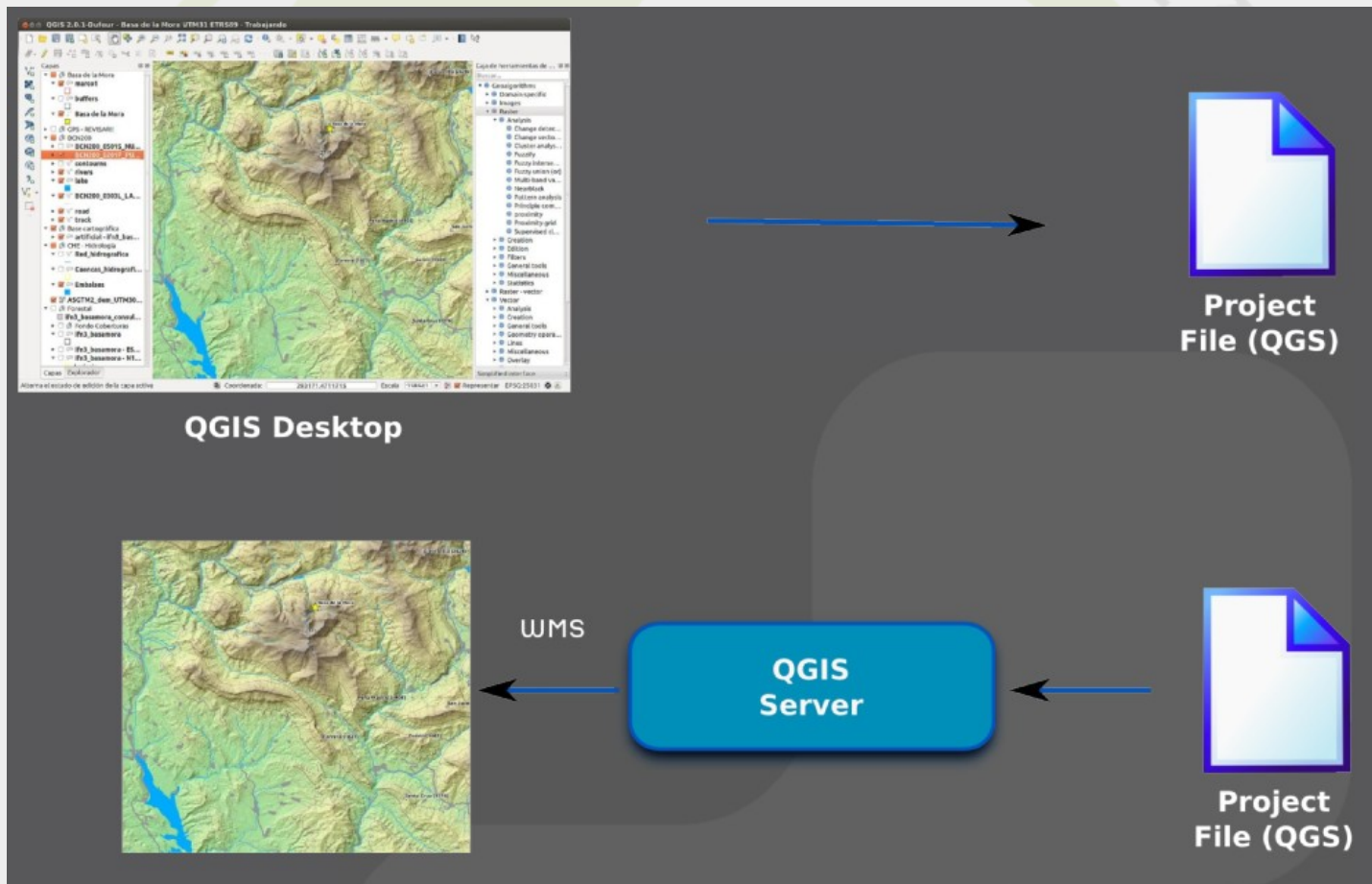
- Annoncé le 11 mai 2007
  - « Dear QGIS developer and users I'm happy to announce the start of the 'QGIS mapserver' project ... The idea of QGIS mapserver is simple: instead of using QGIS just as a desktop GIS, it can also be used as a server. The benefit is that bug fixes and extensions for the server also improve the desktop GIS (and the other way round) ... Contact me if you are interesting in joining development of QGIS mapserver, there is still a lot to do... »

# Origines de QGIS Server

- Utiliser QGIS comme un moteur de rendu



# QGIS Server





# QGIS Server

- Partie intégrante de QGIS depuis 2010
-  • Web Map Service 1.3.0
- Ajout du WMS 1.1.1 en 2012
-  • Ajout du Web Feature Service 1.0.0 en 2012
- Ajout du Transactional WFS 1.0.0 en 2012
-  • Ajout du Web Coverage Service 1.0.0 en 2013
- QGIS Mapserver => QGIS Server

# QGIS Server



# QGIS Server

- Des extensions
  - WMS GetMap au format DXF
  - WMS GetPrint
  - WMS selection, filtre, etc
  - WFS Filtre par expression
  - WFS Simplification des géométries (centroid, extent, none)

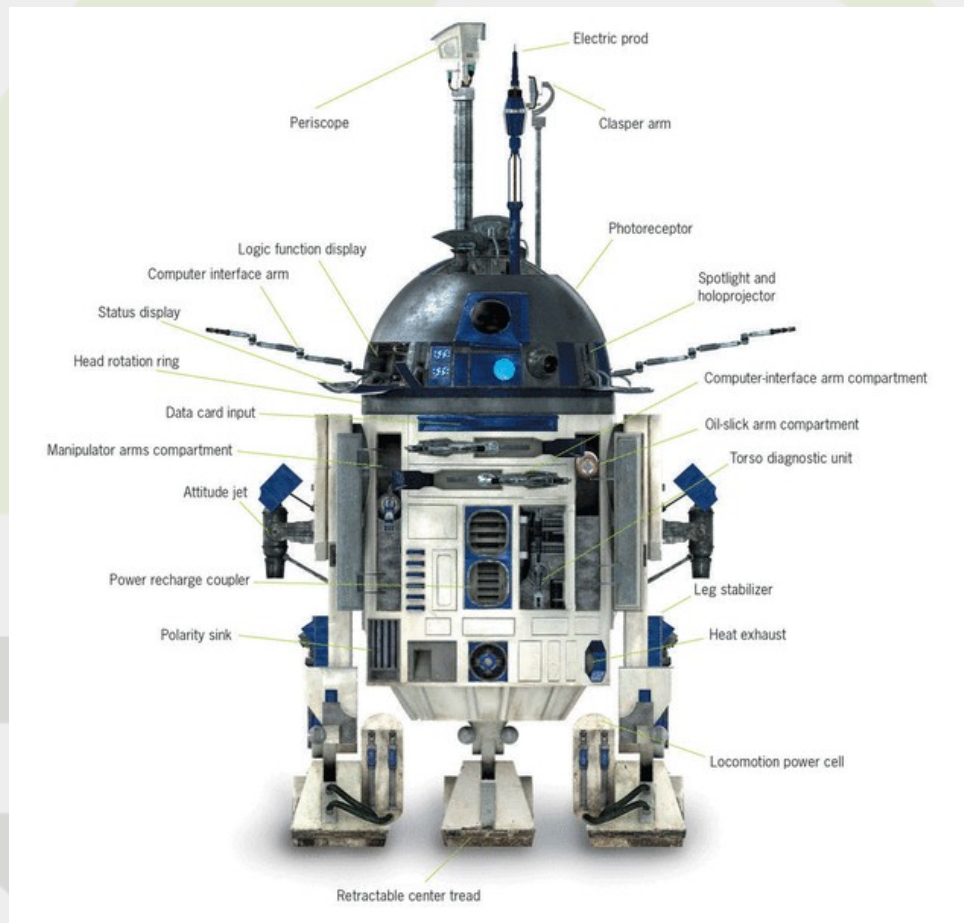


# Un apport mutuel

- Heatmap, blend mode, exportToGeoJSON...

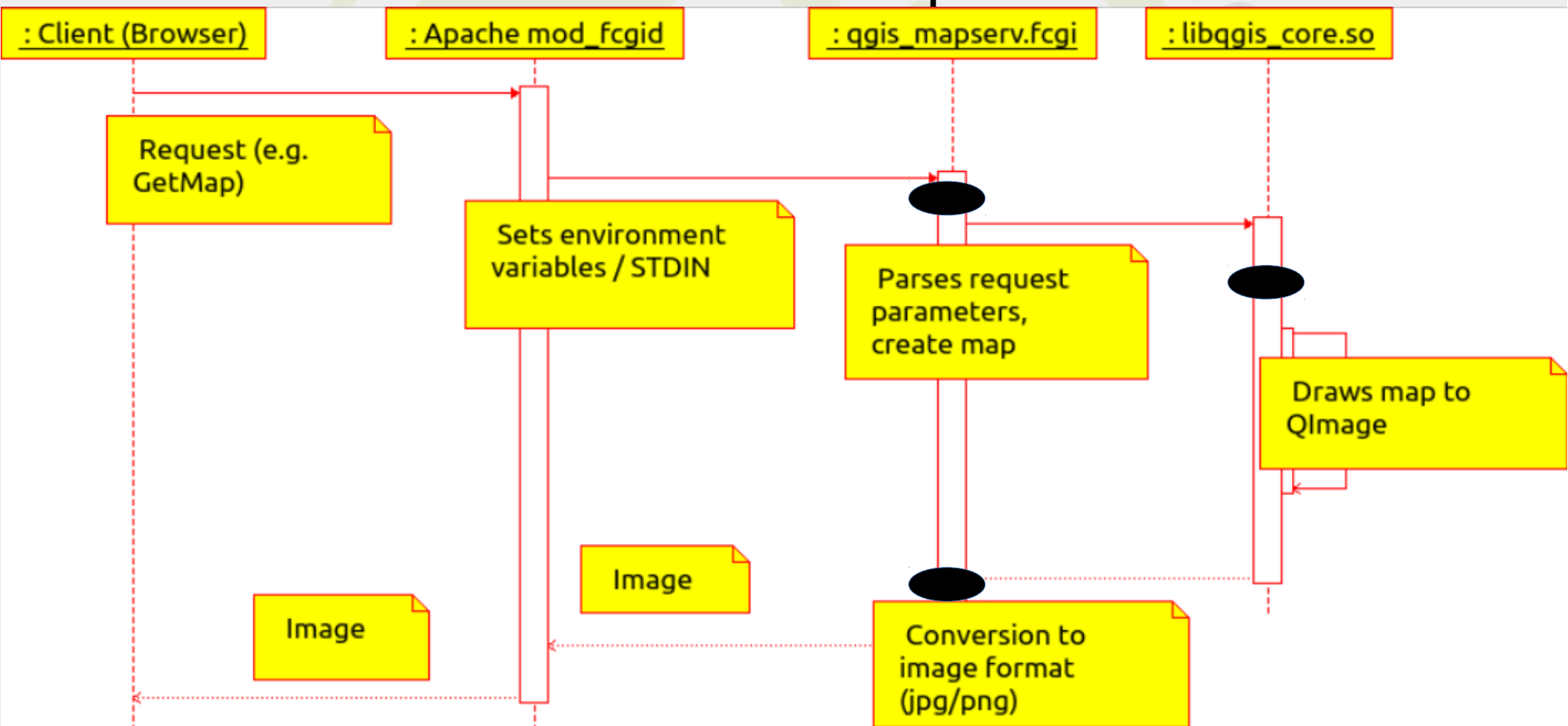


# Les Extensions Python



# Les extensions Python

- Prendre le contrôle des requêtes



# Les Extensions Python

- Depuis QGIS 2.8
- Prendre le contrôle des requêtes
  - Modifier les paramètres entrant
    - Forcer un paramètre
  - Modifier la réponse
    - Incrusté un filigrane
  - Contrôler les données
- Ajouter de nouveaux services (standards ou non)



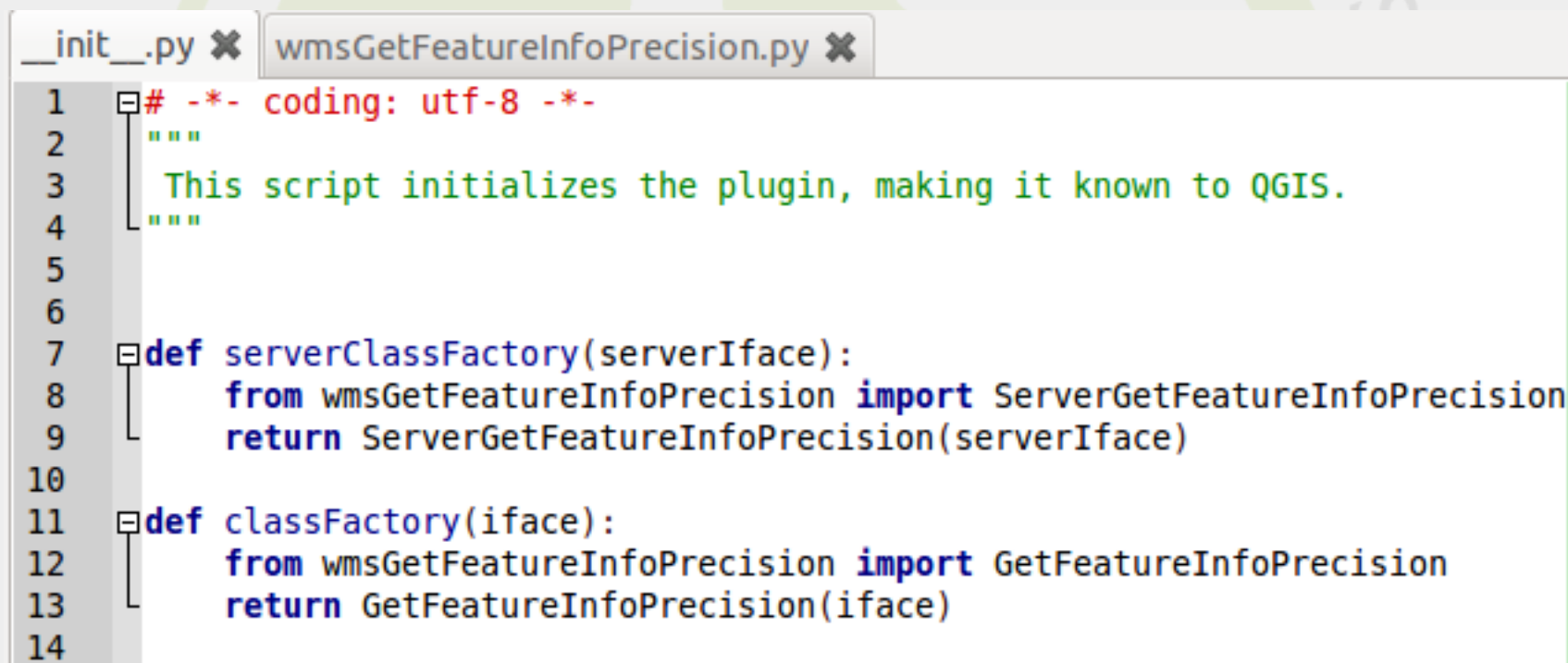
# Les Extensions Python

- wmsGetFeatureInfoPrecision
  - Améliorer WMS GetFeatureInfo
  - Ajouter les paramètres
    - FI\_POINT\_TOLERANCE
    - FI\_LINE\_TOLERANCE
    - FI\_POLYGON\_TOLERANCE
  - Depuis QGIS 2.10



# Les Extensions Python

- wmsGetFeatureInfoPrecision



```
__init__.py ✕ wmsGetFeatureInfoPrecision.py ✕
1  # -*- coding: utf-8 -*-
2  """
3  This script initializes the plugin, making it known to QGIS.
4  """
5
6
7  def serverClassFactory(serverIface):
8      from wmsGetFeatureInfoPrecision import ServerGetFeatureInfoPrecision
9      return ServerGetFeatureInfoPrecision(serverIface)
10
11  def classFactory(iface):
12      from wmsGetFeatureInfoPrecision import GetFeatureInfoPrecision
13      return GetFeatureInfoPrecision(iface)
14
```

# Les Extensions Python

- wmsGetFeatureInfoPrecision

```
64 class ServerGetFeatureInfoPrecision:
65     """Plugin for QGIS server"""
66
67     def __init__(self, serverIface):
68         # Save reference to the QGIS server interface
69         self.serverIface = serverIface
70         try:
71             self.serverIface.registerFilter(ServerGetFeatureInfoPrecisionFilter(serverIface), 1000)
72         except Exception, e:
73             QgsLogger.debug("ServerGetFeatureInfoPrecision- Error loading filter %s", e)
74
```

# Les Extensions Python

```
33 FI_POINT_TOLERANCE = 16
34 FI_LINE_TOLERANCE = 8
35 FI_POLYGON_TOLERANCE = 4
36
37 class ServerGetFeatureInfoPrecisionFilter(QgsServerFilter):
38
39     def requestReady(self):
40         request = self.serverInterface().requestHandler()
41         params = request.parameterMap( )
42         if params.get('SERVICE', '').lower() == 'wms' \
43             and params.get('REQUEST', '').lower() == 'getfeatureinfo':
44             # Test config file
45             if os.path.exists(os.path.join(os.path.dirname(os.path.realpath(__file__)), 'config.cfg')):
46                 config = ConfigParser.ConfigParser()
47                 config.read(os.path.join(os.path.dirname(os.path.realpath(__file__)), 'config.cfg'))
48
49                 pointTolerance = config.get('default', 'FI_POINT_TOLERANCE', str(FI_POINT_TOLERANCE))
50                 request.setParameter('FI_POINT_TOLERANCE', str(pointTolerance))
51
52                 lineTolerance = config.get('default', 'FI_LINE_TOLERANCE', str(FI_LINE_TOLERANCE))
53                 request.setParameter('FI_LINE_TOLERANCE', str(lineTolerance))
54
55                 polygonTolerance = config.get('default', 'FI_POLYGON_TOLERANCE', str(FI_POLYGON_TOLERANCE))
56                 request.setParameter('FI_POLYGON_TOLERANCE', str(polygonTolerance))
57             else:
58                 request.setParameter('FI_POINT_TOLERANCE', str(FI_POINT_TOLERANCE))
59                 request.setParameter('FI_LINE_TOLERANCE', str(FI_LINE_TOLERANCE))
60                 request.setParameter('FI_POLYGON_TOLERANCE', str(FI_POLYGON_TOLERANCE))
61
62
```

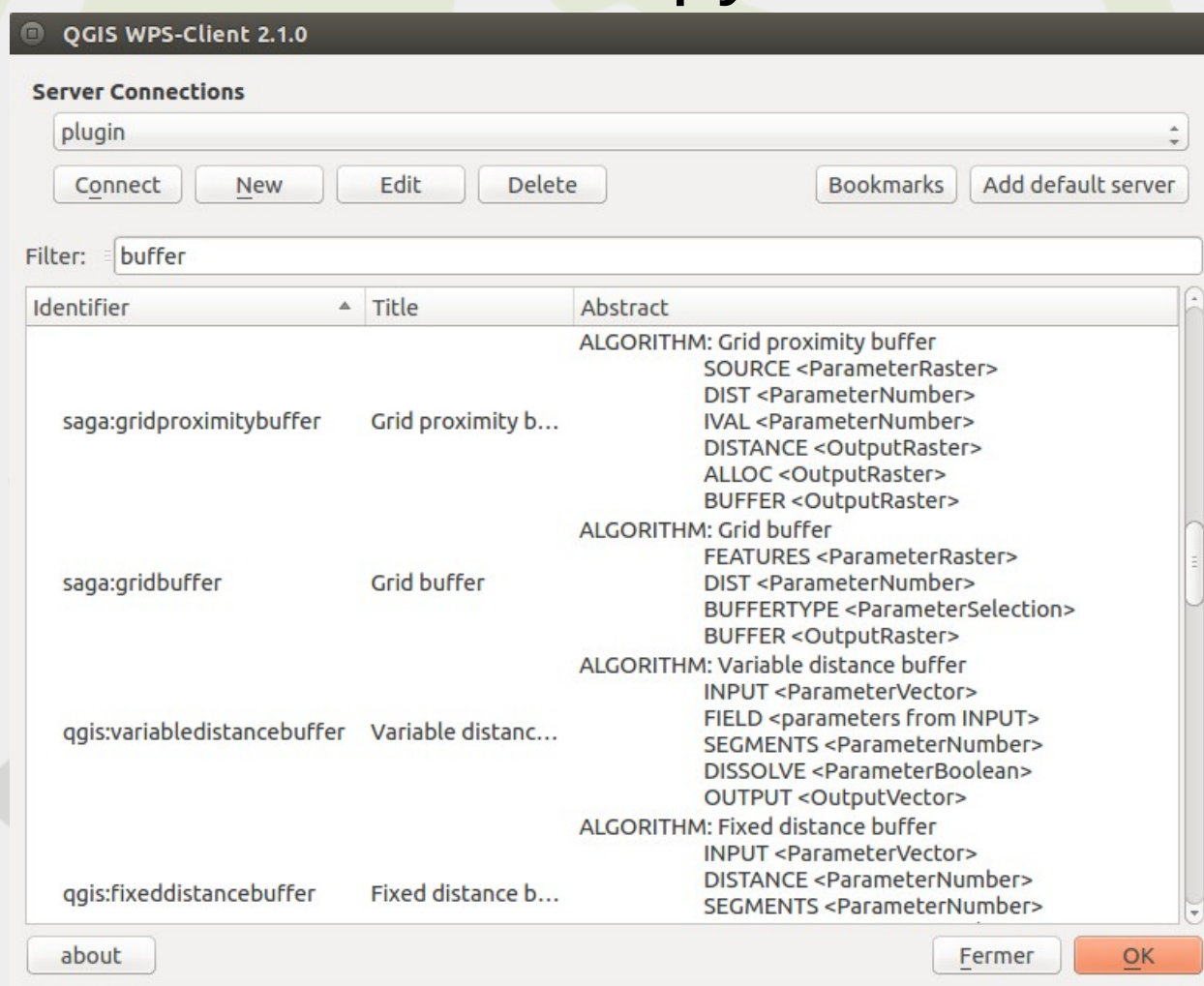
# Les Extensions Python

- wmsGetFeatureInfoPrecision

```
_init_.py ✕ wmsGetFeatureInfoPrecision.py ✕ metadata.txt ✕
1 [general]
2 name=QGIS Server WMS GetFeatureInfo Precision
3 qgisMinimumVersion=2.10
4 qgisMaximumVersion=2.99
5 description=Set WMS GetFeatureInfo Precision
6 version=1.0
7 author=DHONT René-Luc (3Liz)
8 email=rldhont@3Liz.com
9 ; if True it's a server plugin
10 server=True
11
12 about=wmsGetFeatureInfoPrecision adds precision parameters to WMS GetFeatureInfo Request.
13
14 tracker=https://github.com/3liz/qgis-wmsGetFeatureInfoPrecision
15 repository=https://github.com/3liz/qgis-wmsGetFeatureInfoPrecision
16 # End of mandatory metadata
17
18 # Recommended items:
19
20 # Uncomment the following line and add your changelog:
21 # changelog=
22
23 external_deps=none,really
24
25 # Tags are comma separated with spaces allowed
26 tags=server, wms, precision
27
28 homepage=https://github.com/3liz/qgis-wmsGetFeatureInfoPrecision
29 category=server
30 icon=icon.png
```

# Les Extensions Python

- wps4server : Web Processing Service basé sur le module Traitement et pyWPS



# Les Extensions Python

- wps4server : Web Processing Service

```
509 class wpsFilter(QgsServerFilter):
510
511     def __init__(self, serverIface):
512         super(wpsFilter, self).__init__(serverIface)
513
514     def requestReady(self):
515         """request ready"""
516         #QgsMessageLog.logMessage("wpsFilter.requestReady")
517
518
519     def sendResponse(self):
520         """send response"""
521         #QgsMessageLog.logMessage("wpsFilter.sendResponse")
522
523     def responseComplete(self):
524         QgsMessageLog.logMessage("wpsFilter.responseComplete")
525         request = self.serverInterface().requestHandler()
526         params = request.parameterMap()
527         service = params.get('SERVICE', '')
528         if service and service.upper() == 'WPS':
529             # prepare query
530             inputQuery = '&'.join(["%s=%s" % (k, params[k]) for k in params if k.lower() != 'm
531             request_body = params.get('REQUEST_BODY', '')
532
```



# Les Extensions Python

- wfsOutputExtension

```
-<GetFeature>  
  -<ResultFormat>  
    <GML2/>  
    <GML3/>  
    <GeoJSON/>  
    <SHP/>  
    <XLSX/>  
    <ODS/>  
    <KML/>  
    <MIF/>  
    <TAB/>  
    <CSV/>  
  </ResultFormat>  
-<DCPType>
```

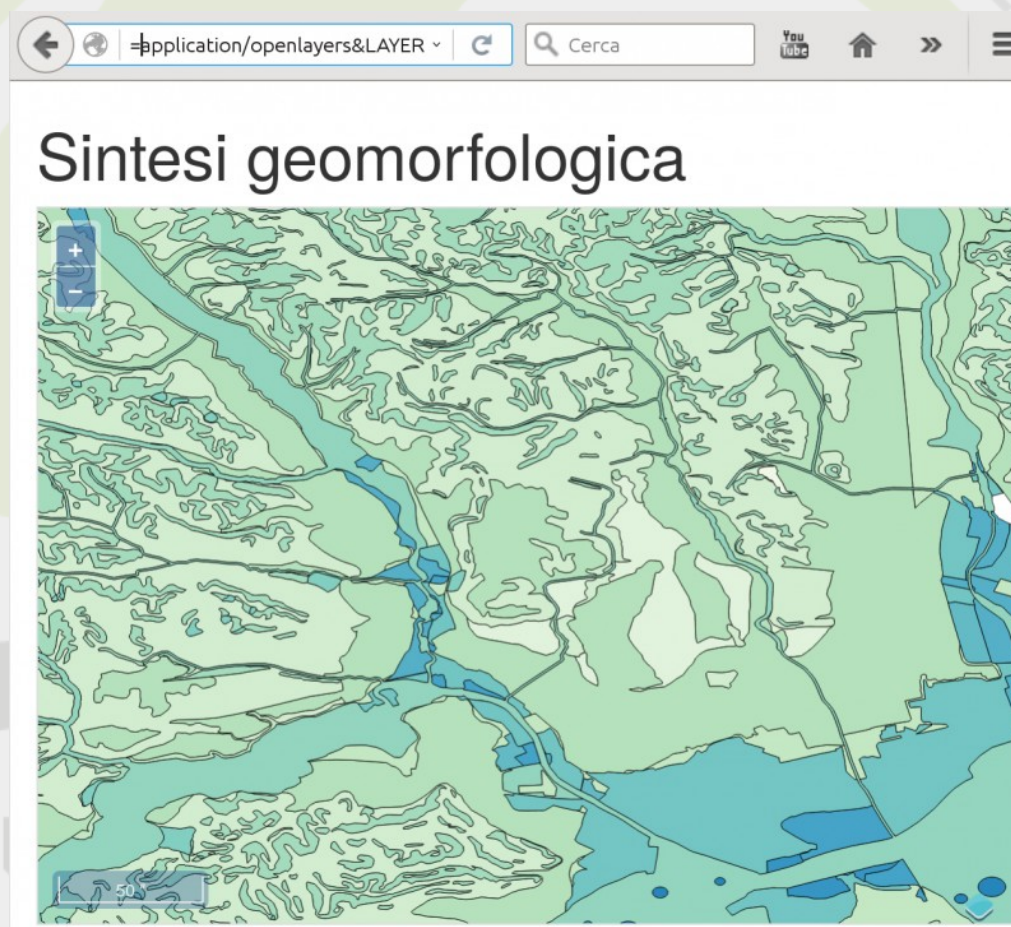
# Les Extensions Python

- wfsOutputExtension

```
94 class WFSFilter(QgsServerFilter):
95
96     def __init__(self, serverIface):
97         QgsMessageLog.logMessage("WFSFilter.init")
98         super(WFSFilter, self).__init__(serverIface)
99         self.format = None
100         self.typename = ""
101         self.filename = ""
102
103         self.tempdir = os.path.join( tempfile.gettempdir(), 'qgis_wfs' )
104         if not os.path.exists(self.tempdir):
105             os.mkdir( self.tempdir )
106         QgsMessageLog.logMessage("WFSFilter.tempdir: %s" % self.tempdir)
107
108     def requestReady(self):
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131     def sendResponse(self):
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     def responseComplete(self):
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
```



# Les Extensions Python

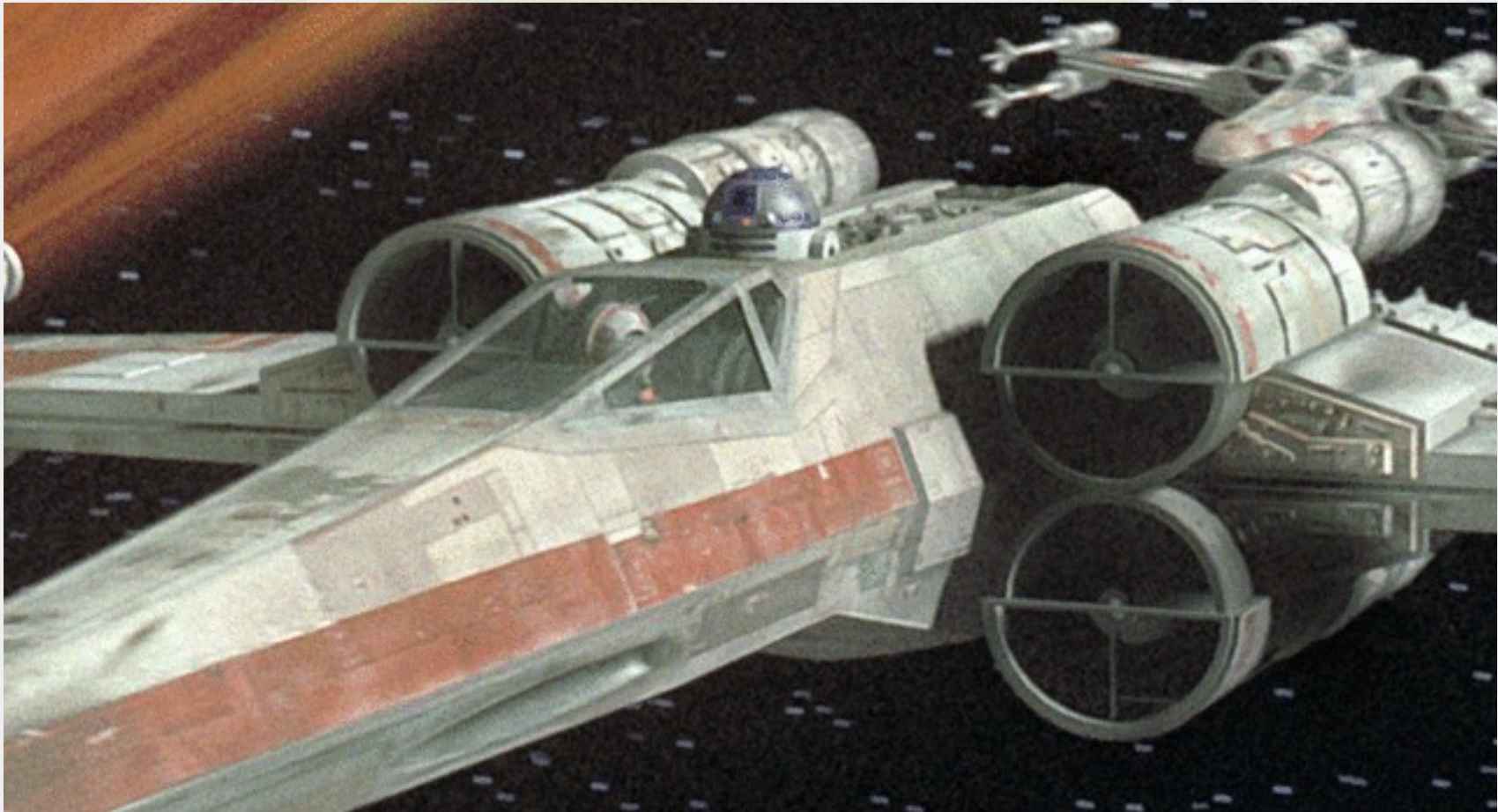


# Les extensions Python

- Contrôle de l'accès aux données

```
82 class RestrictedAccessControl(QgsAccessControlFilter):
83
84     """ Used to have restriction access """
85
86     # Be able to deactivate the access control to have a reference point
87     _active = False
88
89     def init (self, server iface):
90
91     def layerFilterExpression(self, layer):
92
93     def layerFilterSubsetString(self, layer):
94
95     def layerPermissions(self, layer):
96
97     def authorizedLayerAttributes(self, layer, attributes):
98
99     def allowToEdit(self, layer, feature):
100
101     def cacheKey(self):
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 server = QgsServer()
159 server.handleRequest()
160 server_iface = server.serverInterface()
161 access_control = RestrictedAccessControl(server_iface)
162 server_iface.registerAccessControl(access_control, 100)
163
```

# L'API qgis.server



# L'API qgis.server

- Depuis QGIS 2.12
- Faciliter la création de tests
- Embarquer QGIS Server



# L'API qgis.server

```
test_qgsserver.py ✕
1  # -*- coding: utf-8 -*-
2  """QGIS Unit tests for QgsServer.
3
4  .. note:: This program is free software; you can redistribute it and/or modify
5  it under the terms of the GNU General Public License as published by
6  the Free Software Foundation; either version 2 of the License, or
7  (at your option) any later version.
8  """
9  __author__ = 'Alessandro Pasotti'
10 __date__ = '25/05/2015'
11 __copyright__ = 'Copyright 2015, The QGIS Project'
12 # This will get replaced with a git SHA1 when you do a git archive
13 __revision__ = '$Format:%H$'
14
15 import os
16 import re
17 import unittest
18 import urllib
19 from qgis.server import QgsServer
20 from qgis.core import QgsMessageLog
21 from utilities import unitTestDataPath
22
23 # Strip path and content length because path may vary
24 RE_STRIP_PATH = r'MAP=[^&]+|Content-Length: \d+'
25
26
27 class TestQgsServer(unittest.TestCase):
```

# L'API qgis.server

```
82 class RestrictedAccessControl(QgsAccessControlFilter):
83
84     """ Used to have restriction access """
85
86     # Be able to deactivate the access control to have a reference point
87     _active = False
88
89     def init (self, server iface):
90
91
92     def layerFilterExpression(self, layer):
93
94
95
96
97
98
99
100    def layerFilterSubsetString(self, layer):
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115    def layerPermissions(self, layer):
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136    def authorizedLayerAttributes(self, layer, attributes):
137
138
139
140
141
142
143
144
145
146    def allowToEdit(self, layer, feature):
147
148
149
150
151
152
153
154    def cacheKey(self):
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
```

# L'API qgis.server

```
144 def wms_request_compare(self, request, extra=None, reference_file=None):
145     project = self.testdata_path + "test+project.qgs"
146     assert os.path.exists(project), "Project file not found: " + project
147
148     query_string = 'MAP=%s&SERVICE=WMS&VERSION=1.3&REQUEST=%s' % (urllib.quote(project), request)
149     if extra is not None:
150         query_string += extra
151     header, body = [str(_v) for _v in self.server.handleRequest(query_string)]
152     response = header + body
153     f = open(self.testdata_path + (request.lower() if not reference_file else reference_file) + '.txt')
154     expected = f.read()
155     f.close()
156     # Store the output for debug or to regenerate the reference documents:
157     """
158     f = open(os.path.dirname(__file__) + '/expected.txt', 'w+')
159     f.write(expected)
160     f.close()
161     f = open(os.path.dirname(__file__) + '/response.txt', 'w+')
162     f.write(response)
163     f.close()
164     """
165     response = re.sub(RE_STRIP_PATH, '', response)
166     expected = re.sub(RE_STRIP_PATH, '', expected)
167
168     # for older GDAL versions (<2.0), id field will be integer type
169     if int(osgeo.gdal.VersionInfo()[1]) < 2:
170         expected = expected.replace('typeName="Integer64" precision="0" length="10" editType="TextEdit" type')
171
172     self.assertEqual(response, expected, msg="request %s failed.\n Query: %s\n Expected:\n%s\n\n Response:\n")
173
```

# L'API qgis.server

- `python/server/qgsserverinterface.sip`
- `python/server/qgsserverfilter.sip`
- `python/server/qgsaccesscontrolfilter.sip`
- `tests/src/python/test_qgsserver.py`
- `tests/src/python/test_qgsserver_accesscontrol.py`



# L'API qgis.server

```
3. Super simple QgsServer.
4. """
5.
6. from qgis.server import *
7. from BaseHTTPServer import *
8.
9. class handler (BaseHTTPRequestHandler):
10.
11.     server = QgsServer()
12.
13.     def doHeaders(self, response):
14.         l = response.pop(0)
15.         while l:
16.             h = l.split(':')
17.             self.send_header(h[0], ' '.join(h[1:]))
18.             self.log_message( "send_header %s - %s" % (h[0], ' '.join(h[1:])) )
19.             l = response.pop(0)
20.         self.end_headers()
21.
22.     def do_HEAD(self):
23.         self.send_response(200)
24.         response = str(handler.server.handleRequestGetHeaders(self.path[2:])).split(
25.             self._doHeaders(response)
26.
27.     def do_GET(self):
28.         response = str(handler.server.handleRequest(self.path[2:])).split('\n')
29.         i = 0
30.         self.send_response(200)
31.         self._doHeaders(response)
32.         self.wfile.write((' \n'.join(response[i:])).strip())
33.
34.     def do_OPTIONS(s):
35.         handler.do_GET(s)
36.
37. httpd = HTTPServer( ('', 8000), handler)
```

# L'API qgis server

```
1. # QGIS server view
2.
3. from django.http import HttpResponse
4. from django.views.generic import View
5. from qgis.server import *
6.
7.
8. class OGC(View):
9.     """Pass a GET request to QGIS Server and return the response"""
10.
11.     def __init__(self):
12.         self.server = QgsServer()
13.
14.     def get(self, request, *args, **kwargs):
15.         """Pass a GET request to QGIS Server and return the response"""
16.         headers, body = self.server.handleRequest(request.GET.urlencode())
17.         response = HttpResponse(body)
18.         # Parse headers
19.         for header in headers.split('\n'):
20.             if header:
21.                 k, v = header.split(': ', 1)
22.                 response[k] = v
23.         return response
24.
```

# L'API qgis.server



# L'avenir de QGIS Server



# L'avenir de QGIS Server

- ~~Meilleur respect de la norme ISO~~
- ~~Faciliter la saisie des propriétés~~
- Plugin Server WMTS
- ~~Outil Bureautique de validation~~
- ~~ShowFeatureCount pour GetLegendGraphic~~
- ~~WMS INSPIRE~~
- WFS 2 ?

# FOSS4G-fr 2016

Merci !

Des Questions ?