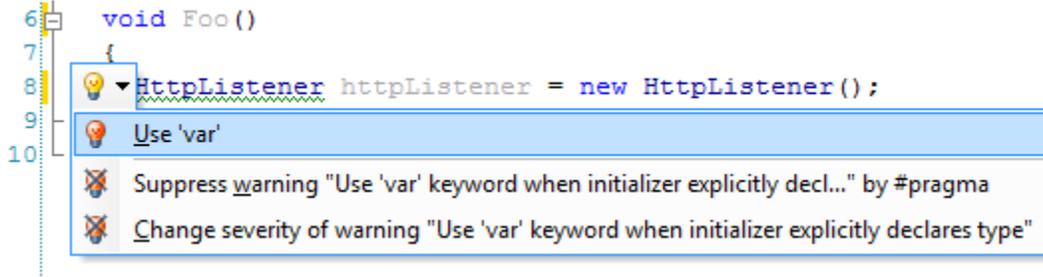


ReSharper 4.0 EAP Notes

C# 3.0 support

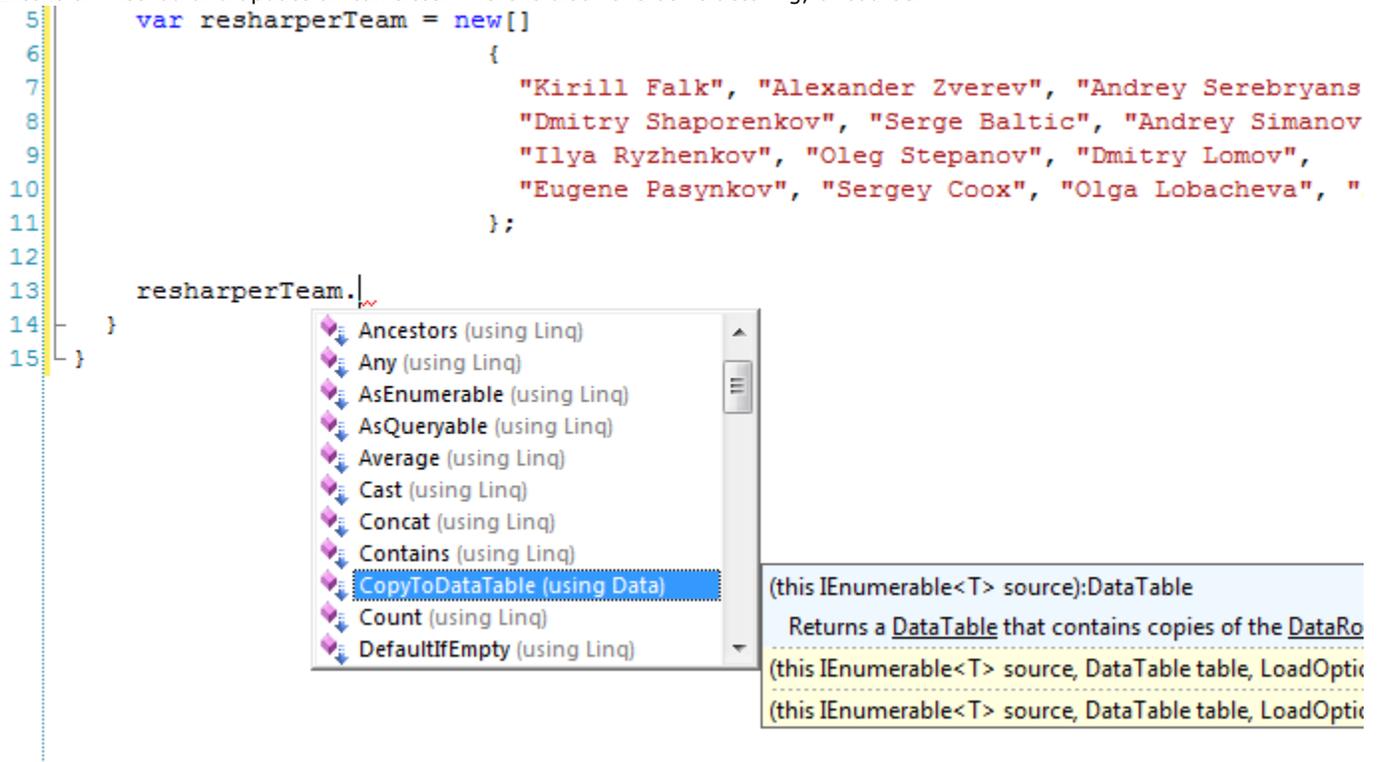
Implicitly typed locals and arrays

Fully supported, including suggestions to omit explicit type specification, Introduce Variable refactoring and context actions to convert to implicit or explicit form.



Extension Methods support

Fully supported, with Import Symbol completion (previously known as Type Name completion) after dot which can import suitable extension method and insert appropriate using directive. Analysis will suggest converting invocations from static-like to instance-like form. Context action can convert it back, and there are refactorings to convert static method of static class to Extension Method and update all call sites. There is also reverse refactoring, of course.



Object and collection initializers

Fully supported, with suggestions and context actions to fold series of statements to collection or object to initializer, if possible.

```

14 var person = new Mathematician();
15
16 Use object initializer
17
18 Inline as anonymous type
19
20 Split declaration and assignment
21
22 Suppress warning "Use object or collection initializer when possible" by #pragma
23
24 Change severity of warning "Use object or collection initializer when possible"
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

```

ReSharper also provides smart intellisense features in such places, like completing settable properties or displaying correct parameter information for Add method used in collection initializer.

```

18 [string FirstName], [string LastName], [DateTime BirthDate], [string Nationality], [Uri PageUri], [ICollection<string> Res
19 {
20     FirstName = "Haskell",
21     LastName = "Curry",
22     BirthDate = new DateTime(1900, 9, 12),
23
24     }
25 }
26 }

```

Automatic properties

Fully supported, with context actions to convert to property with backing field and back. Analysis can show properties which are candidates for conversion. Code generation features may suggest creating automatic property in addition to property with backing field, if appropriate.

```

7 public string FirstName { get { return _firstName; } set { _firstName = value; }
8
9
10
11
12

```

Anonymous types

Fully supported. ReSharper can search for similar anonymous types throughout the project or solution, give a name to anonymous type and replace usages, rename properties, suggest reordering properties if two anonymous types look similar and more.

```

5     var ranges = new[]
6         {
7             new {From = 0, To = 2},
8             new {From = 3, To = 4},
9             new {From = 7, To = 12}
10        };
11
12     var rangeToSearch = new {To = 5, From = 4};
13
14

```

Lambdas

Partially supported. By partially, it means that not all features we want there are ready. The code shouldn't be red and intellisense should be working, but some context actions, quickfixes, analysis and such may be missing. We are actively working on those features, so as time passes ReSharper will be more and more smart about lambdas. We have conversion actions like delegate to lambda and back and some suggestions. Note, that not all features of ReSharper are lambda-aware yet, so they may fail when they happen to operate on lambdas.

```

9     var names = new[] { "George", "Michael", "Tommy" };
10    if (names.Any(n => n.StartsWith("G")))
11    {
12        var nonEmpty = names.Where(delegate(string arg)
13            {
14                return arg.Length > 0;
15            });
16    }
17
18
19

```

LINQ

Fully supported. We import appropriate extension methods, provide intellisense, a number quickfixes, our refactorings are aware of queries. We also provide special live templates.

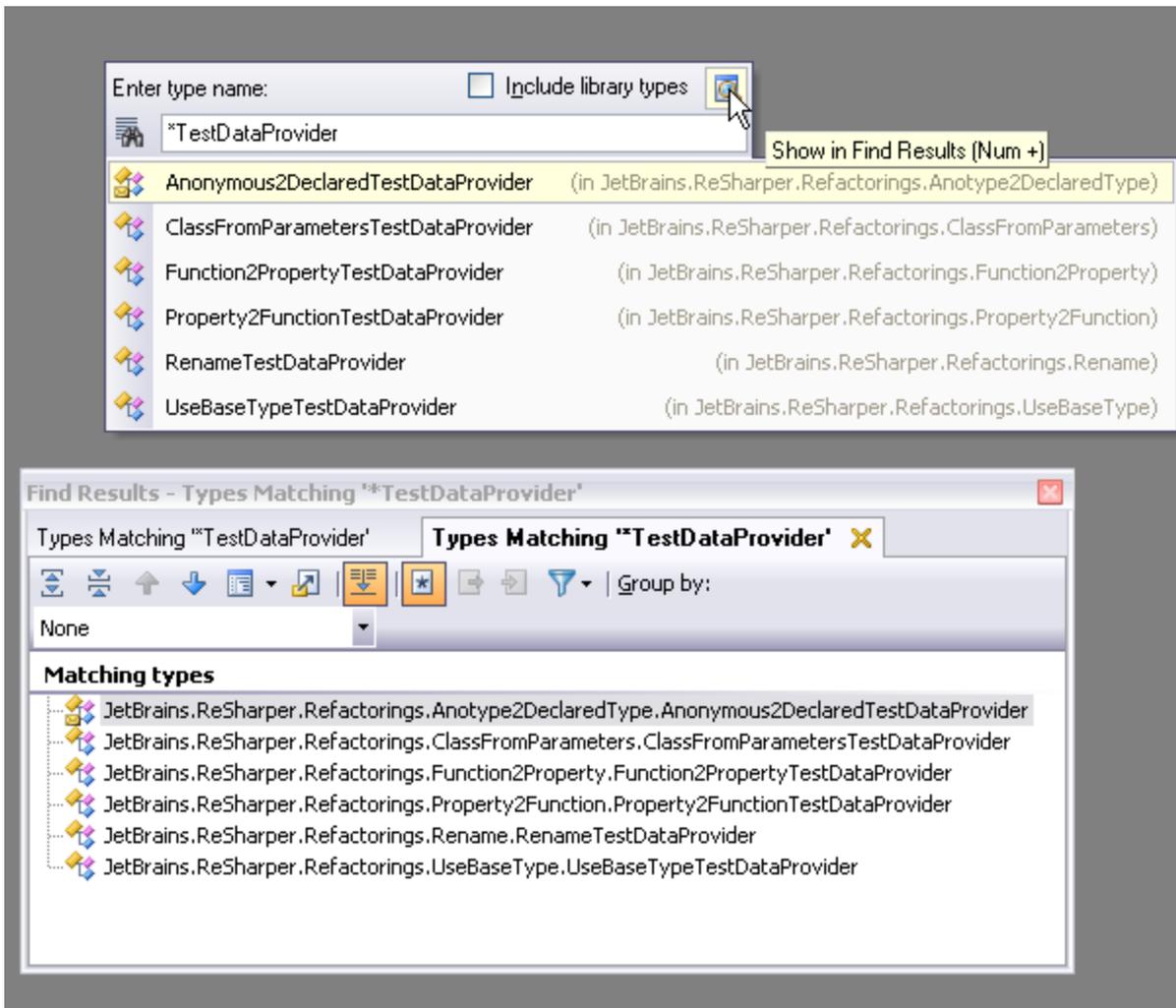
Partial methods

Fully supported.

New and updated features

Go to Type, File, Symbol and Find Results

You can now put results of "Go to" features to find results for browsing



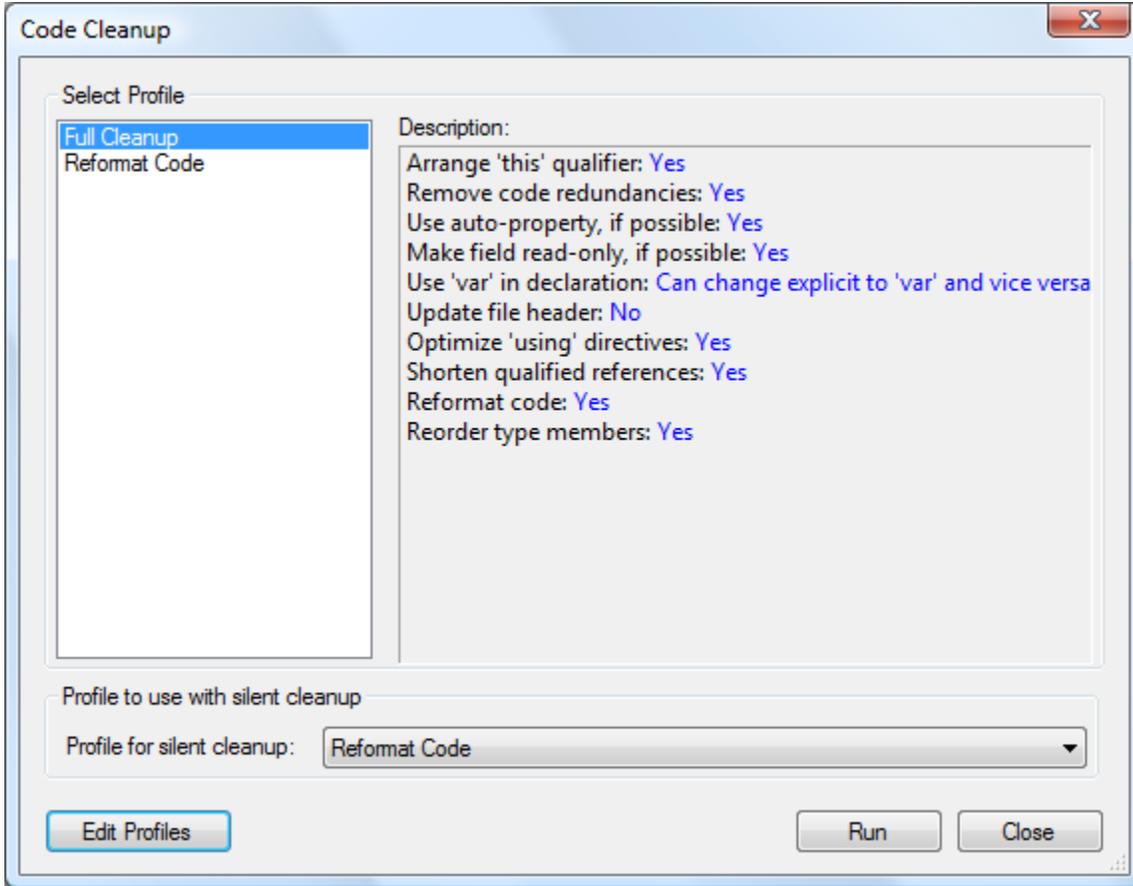
External Annotations

Ability to annotate external (non-source) symbols with ReSharper-specific attributes, like CanBeNull and NotNull. Note, that we standardized attribute names for better interoperability and less configuration hassle, so you may need to refactor you code a bit if you use these attributes.

Code Cleanup

Successor of Reformat Code, Optimize Usings and other code cleaning features - all in one interface with many more modules. Includes things like processing readonly fields, removing redundancies, updating file header, converting to automatic

properties, replacing explicit types with vars and more. All in batch mode, so that you can instantly clean the whole project or even solution.



Plenty of new analyses

We added many more inspections in ReSharper 4. Some of them are related to C# 3.0 and deal with new language constructs. Also, there are new structure-related inspections, like "if" statement analysis, anonymous method closure analysis, and more. We also added new severity level for analysis results - Hint - which is a lot less intrusive. It does not participate in next/previous highlight navigation and is not shown on error stripe. It simply tells you: "Pssst! Take a look here, it can make your code look better."

Complete Statement

Also known as Smart Enter, it inserts required syntax elements to complete the code you are writing. It can close parentheses as needed, add semicolon, complete constructs such as if, while, for and so on.

```
public static Processor CreateProcessor(Context context)
{
    if| return new Processor(context);
}

public static P
{
    if ()
    {
        return new
    }
}
```

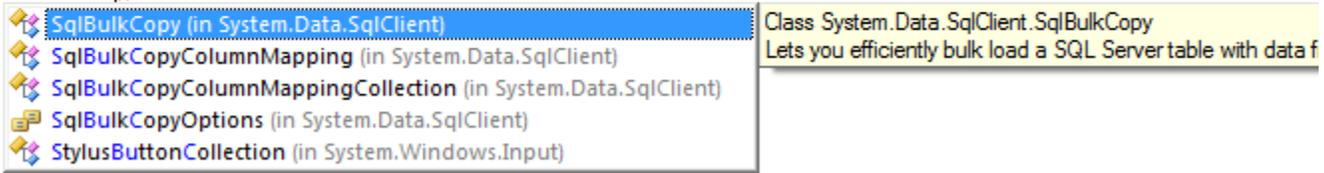
Completion with CamelHumps

All kinds of code completion (Symbol completion, Import Symbol Completion and Smart Completion) now supports filtering by CamelHumps, which tremendously speeds things up. E.g. if you know you have DefaultCodeCompletionManager type

somewhere you just type DCCM and invoke Import symbol completion. Full type name will be inserted and using directive will be added as needed.

```
namespace SampleApplication
{
    public class Processor
    {
        public static Processor CreateProcessor(Context context)
        {
            SBuC

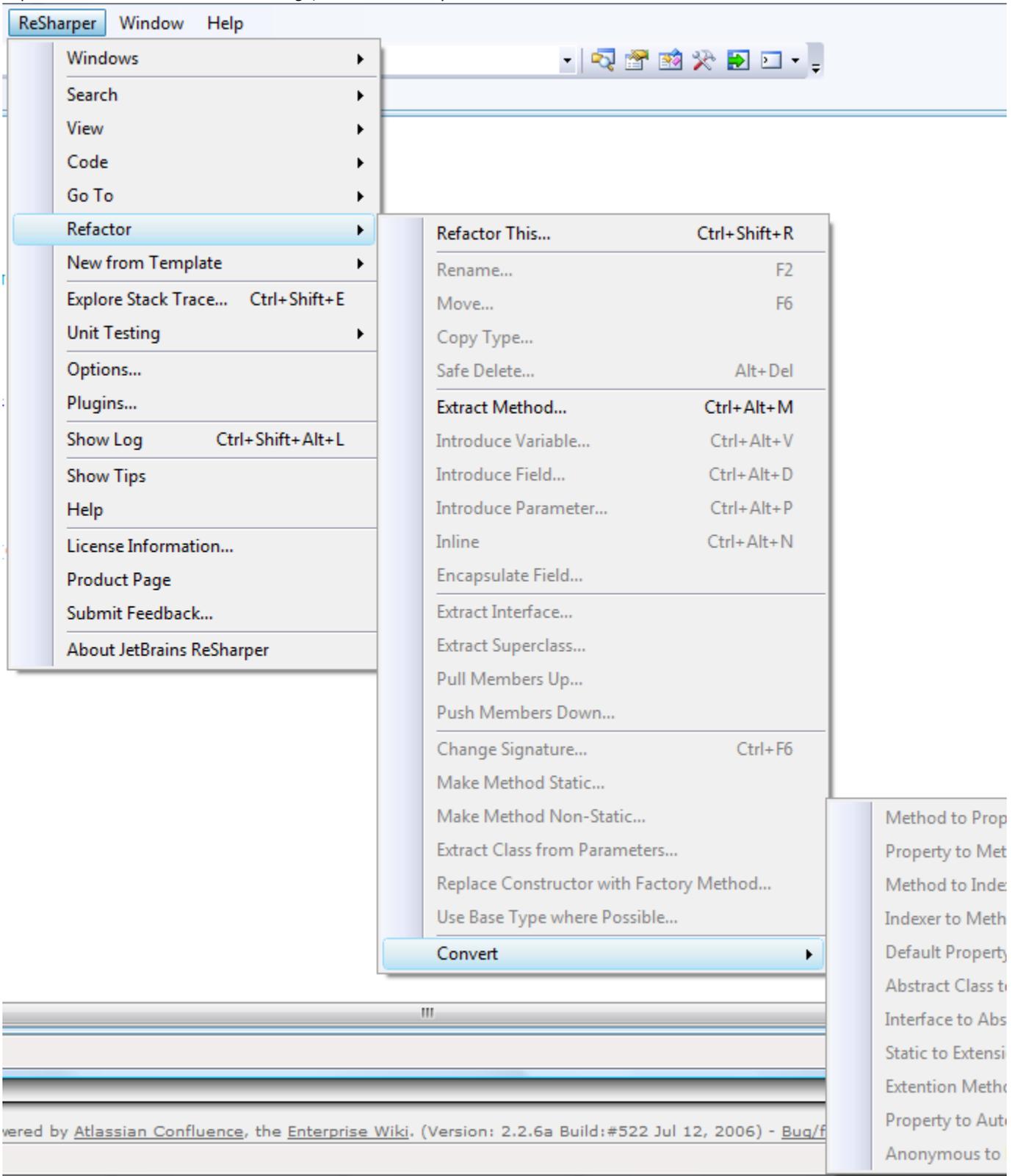
```



Refactorings

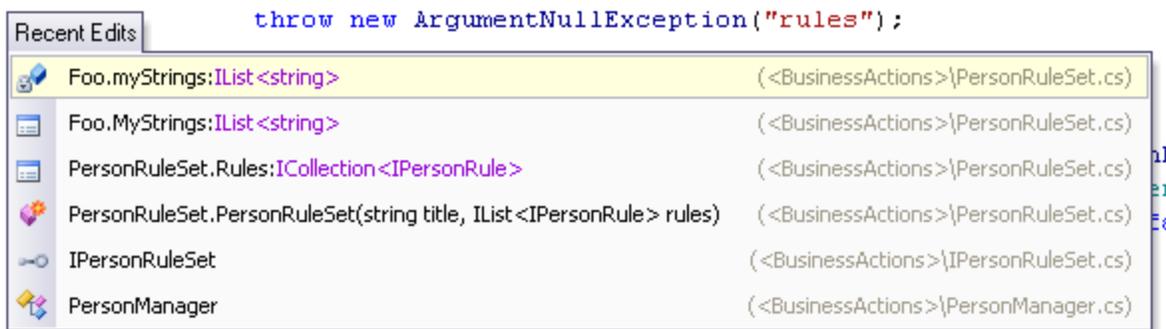
There are new refactorings related to C# 3.0 features, like "Convert Property to Automatic Property" and back, "Convert Static Method to Extension Method" and back and "Name Anonymous Type". "Introduce Variable" is now dialogless and this improves coding flow a lot! Another great new refactoring is "Inline Method", which can replace method call with method body at specific

point or all over your code. "Method to Indexer" is a nice addition to "Method to Property" refactoring. There are also improvements in almost all refactorings, both in usability and smartness.



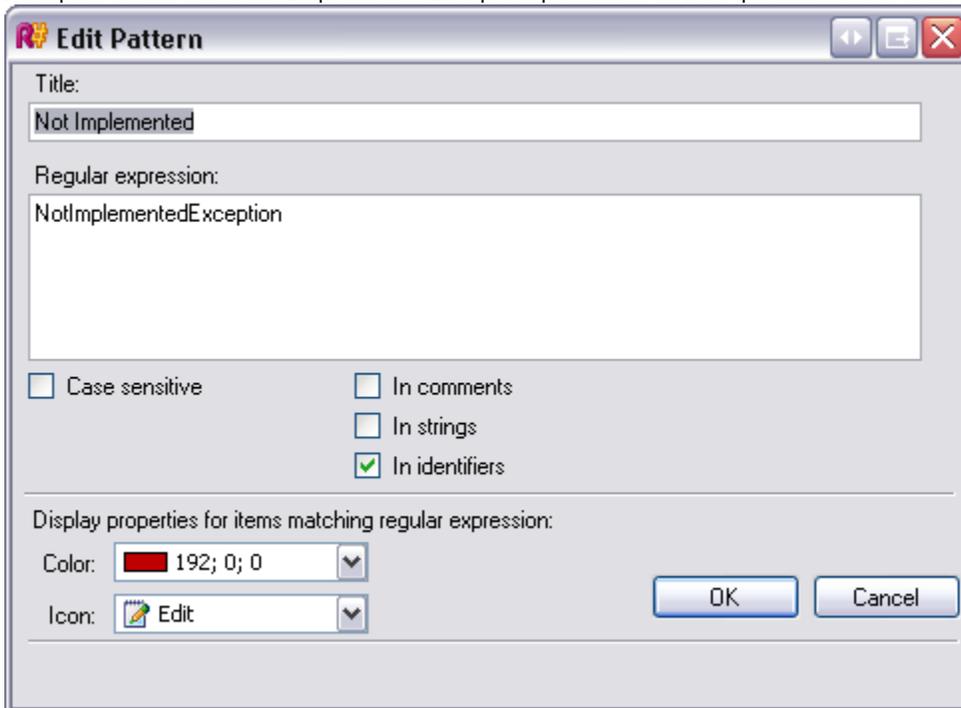
Recent Edits

In addition to Recent Files we created Recent Edits feature, which shows what you've been changing recently. If appropriate, it displays type and member the change occurred in.



To-do items on identifiers and string literals

You can now configure ReSharper to search for specific patterns not only in comments, but also in string literals and identifiers. It is quite useful to set `NotImplementedException` pattern to have a quick access to code you have to write.



Support for solutions using different targets

ReSharper now fully supports solutions where different projects use different versions of CLR like .NET Compact Framework or Silverlight. You can now have e.g. Smart Device projects and normal projects in the same solution and all ReSharper features will work perfectly fine in this scenario.

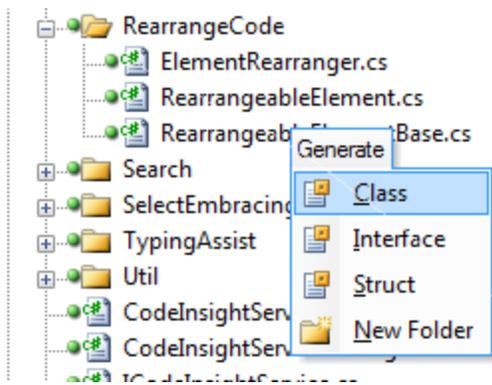
Project References

This is more like preview of the feature which may or may not come into final bits of ReSharper 4. It can show dependencies between projects and libraries in terms of assembly references.

Live Templates Editor

We are improving usability of Live Templates editor and currently it is large work in progress. Be careful with it if you use custom templates.

Easier creation of new folders through generate (Alt+Insert shortcut)



Dozens of other features

We also have dozens of other minor features, keep track of [this tracker query](#) for the current list.