

# ReSharper and Pex

---



Brian Strelloff 85 posts since

Jan 10, 2008

I am using Pex ( <http://www.codeplex.com/Pex> ) to automatically generate test cases for all code paths based on static/dynamic evaluation. Pex generates partial classes, in multiple files. However, RS does not appear to be able to do code cleanup on the files with generated test cases (but it does do cleanup on the generated parameterized unit test files). These files are all in the same directory/project.

Even if I open the generated test case files, the "Code Cleanup" option is disabled (yet other ReSharper options are available). This is with VS2008 and RS EAP 807.

Why would RS exclude some partial class files in the same project? Here is a snippet of the start of a file that is skipped by RS cleanup (Note that RS does flag errors in the file, including global analysis):

```
using Microsoft.Pex.Framework;  
  
using Bks.Framework.Common;  
  
using Microsoft.VisualStudio.TestTools.UnitTesting;  
  
using Microsoft.Pex.Framework.Generated;  
  
using System;  
  
namespace Bks.Framework.Common.Pex  
{  
  
    public partial class CachedStringTest
```

```
{  
  
    [System.Diagnostics.CodeAnalysis.SuppressMessage  
    ( "Microsoft.Naming", "CA1707:IdentifiersShouldNotContainUnderscores" ),  
    System.Diagnostics.CodeAnalysis.SuppressMessage ( "Microsoft.Naming",  
    "CA1709:IdentifiersShouldBeCasedCorrectly", MessageId = "op" ), TestMethod]  
  
    PexGeneratedBy ( typeof ( CachedStringTest ) )  
  
    public void op_GreaterThanCachedStringObject_20080522_220038_000 ( )  
  
    {  
  
        PexValue.Generated.Clear ( );  
  
        this.op_GreaterThan ( ( CachedString ) null, ( object ) null );  
  
        PexValue.Generated.Validate ( "result", "False" );  
  
    }  
}
```

Edited by: Brian Strelloff on May 25, 2008 5:26 PM



Guest 1. **Re: ReSharper and Pex** May 25, 2008 11:47 PM

Hello,

Could you please paste a piece of the .csproj file that includes the file in question? Or make a screenshot of the Solution Explorer with that file selected.

R# fails to reformat files that are parented under other C# files in the solution explorer.

ReSharper and Pex

—

Serge Baltic

JetBrains, Inc — <http://www.jetbrains.com>

“Develop with pleasure!”



[Brian Streliaff](#) *85 posts since*

*Jan 10, 2008* **2. Re: ReSharper and Pex** May 26, 2008 2:28 AM

Here is a snippet from a csproj file:

Note that the `.g.` files are the ones skipped by RS code cleanup.

In the Solution explorer, they show up under their parent (i.e. the "DependentUpon" file via usual tree view display).



Guest **3. Re: ReSharper and Pex** May 26, 2008 4:36 AM

Hello,

ReSharper and Pex

Yes, that's it. R# would currently ignore .cs files that have a DependentUpon metadata item pointing to another .cs file.

Removing this item should not affect compilation but will enable R# analyses and code cleanups. R# regards such files as autogenerated and supposes that they should not be altered. I do not know if there're any plans for adjusting this heuristics.

—

Serge Baltic

JetBrains, Inc — <http://www.jetbrains.com>

“Develop with pleasure!”



[Brian Streliaff](#) *85 posts since*

*Jan 10, 2008* **4. Re: ReSharper and Pex** May 26, 2008 5:35 AM

Maybe add an option on whether or not "DependentUpon" files should be skipped?

I am not sure all users would be satisfied with hardcoding this decision one way or the other.



[Evgeny Pasyukov](#) *6,088 posts since*

*Dec 3, 2003* **5. Re: ReSharper and Pex** May 26, 2008 1:52 PM

ReSharper and Pex

This behaviour is not likely to be modified in ReSharper 4.0

Once we discover more reliable way to tell generated files from manually created, we'll fix this problem

--

Eugene Pasyukov

Developer

JetBrains, Inc

<http://www.jetbrains.com>

"Develop with pleasure!"

"Brian Strelieff" <[BKStrelieff@Hotmail.com](mailto:BKStrelieff@Hotmail.com)> wrote in message

news:[6806554.23271211765773548.JavaMail.jive@app4.labs.intellij.net](mailto:6806554.23271211765773548.JavaMail.jive@app4.labs.intellij.net)...

Maybe add an option on whether or not "DependentUpon" files should be skipped?

>

I am not sure all users would be satisfied with hardcoding this decision one way or the other.



[Gabriel Lozano-Moran](#) 213 posts since

Sep 25, 2007 6. Re: ReSharper and Pex May 26, 2008 1:04 PM

ReSharper and Pex

On a sidenote: I don't recommend using tools like Pex. If you use TDD you should focus on the behaviour of your code and not the implementation. IMHO it is a bad practice to first write your code and then using code coverage tools to write tests for execution paths that are not covered by unit tests. Microsoft should focus on providing tools and frameworks that actually help us do our work and at the same time promote good practices instead of providing us a tool like Pex that actually promotes the bad practices of code first, unit test later.

Regards

Gabriel Lozano-Moran

"Brian Strelhoff" <[BKStrelhoff@Hotmail.com](mailto:BKStrelhoff@Hotmail.com)> wrote in message  
news:[23114470.21611211577422569.JavaMail.jive@app4.labs.intellij.net](mailto:23114470.21611211577422569.JavaMail.jive@app4.labs.intellij.net)...  
>I am using Pex ( <http://www.codeplex.com/Pex> ) to automatically generate  
>test cases for all code paths based on static/dynamic evaluation. Pex  
>generates partial classes, in multiple files. However, RS does not appear  
>to be able to do code cleanup on the files with generated test cases (but  
>it does do cleanup on the generated parameterized unit test files). These  
>files are all in the same directory/project.

>

Even if I open the generated test case files, the "Code Cleanup" option is disabled (yet other ReSharper options are available). This is with VS2008 and RS EAP 807.

>

Why would Pex exclude some partial class files in the same project? Here is a snippet of the start of a file that is skipped by RS cleanup (Note that RS does flag errors in the file, including global analysis):

>

```
using Microsoft.Pex.Framework;  
using Bks.Framework.Common;  
using Microsoft.VisualStudio.TestTools.UnitTesting;  
using Microsoft.Pex.Framework.Generated;  
using System;
```

>

```
namespace Bks.Framework.Common.Pex  
{  
    public partial class CachedStringTest  
    {  
        [System.Diagnostics.CodeAnalysis.SuppressMessage ( "Microsoft.Naming",  
            "CA1707:IdentifiersShouldNotContainUnderscores" ),  
        System.Diagnostics.CodeAnalysis.SuppressMessage ( "Microsoft.Naming",
```

```
"CA1709:IdentifiersShouldBeCasedCorrectly", MessageId = "op" ),  
TestMethod]  
  
PexGeneratedBy ( typeof ( CachedStringTest ) )  
  
public void op_GreaterThanCachedStringObject_20080522_220038_000 ( )  
{  
  
PexValue.Generated.Clear ( );  
  
this.op_GreaterThan ( ( CachedString ) null, ( object ) null );  
  
PexValue.Generated.Validate ( "result", "False" );  
  
}
```



[Brian Strelloff](#) 85 posts since

Jan 10, 2008 7. Re: **ReSharper and Pex** May 26, 2008 5:51 PM

Realistically, while I support TDD there is **\*NEVER\*** enough time or budget allocated for it, nor can the cost of manually generating tests for existing software be justified. Automatic test generators are essential, not only for capturing existing behaviour (i.e. creating regression tests) but also for ensuring timely and cost-effective completeness/coverage of any test suite.

Something like Pex is a far more credible source for tests since it is based on run-time analysis of the code under test (i.e. how the code actually works), rather than potential misunderstanding of potentially missing/incorrect documents (design or usage). There will always be some thought required, for example does any particular test (TDD or generated) capture the proper behaviour, or does it reproduce an existing "unknown" bug behaviour. Similarly, Pex is far more reliable at discovering and writing test cases for "hidden" behaviours/requirements (i.e. those imposed by a subcomponent of the component under test).

TDD is a good idea, but it is not the solution to reliable, affordable, and timely software. Nor is Pex all by itself, or any other technology that I am aware of. But Pex does aid the TDD process (and other areas of software development), and as such reduces the cost while simultaneously increasing the quality of software development.



[Gabriel Lozano-Moran](#) 213 posts since

Sep 25, 2007 8. Re: **ReSharper and Pex** May 26, 2008 7:29 PM

Anyway if you need tools like PEX you are not practicing TDD. I agree that it takes at least a couple of months (2, 3 months) before the developers get the hang of TDD and yes, TDD is the solution to reliable, affordable and timely software. What you will test using PEX is the correctness of your implementation, not the behaviour. PEX will NEVER be adopted by the XP community.

I just said that I don't recommend PEX but if you really want to use it, be my guest and have loads of fun with it.

Using Continuous Integration with a ten-minute build/first-stage build and code coverage tool, you could easily detect when someone checks-in code that was not covered by at least 1 unit test. If this is the case then PEX is still not the answer. I would like to know why this developer checked in code that had no covering unit test and have him/her throw away the code.

This is my 50 cent...

Cheers

Gabriel Lozano-Moran

"Brian Strelieff" <[BKStrelieff@Hotmail.com](mailto:BKStrelieff@Hotmail.com)> wrote in message

news:[8660862.24861211809950226.JavaMail.jive@app4.labs.intellij.net...](mailto:8660862.24861211809950226.JavaMail.jive@app4.labs.intellij.net...)

Realistically, while I support TDD there is **\*NEVER\*** enough time or budget allocated for it, nor can the cost of manually generating tests for existing software be justified. Automatic test generators are essential, not only for capturing existing behaviour (i.e. creating regression tests) but also for ensuring timely and cost-effective completeness/coverage of any test suite.

>

Something like Pex is a far more credible source for tests since it is based on run-time analysis of the code under test (i.e. how the code actually works), rather than potential misunderstanding of potentially missing/incorrect documents (design or usage). There will always be some thought required, for example does any particular test (TDD or generated) capture the proper behaviour, or does it reproduce an existing "unknown" bug behaviour. Similarly, Pex is far more reliable at discovering and writing test cases for "hidden" behaviours/requirements (i.e. those

imposed by a subcomponent of the component under test).

>

TDD is a good idea, but it is not the solution to reliable, affordable, and timely software. Nor is Pex all by itself, or any other technology that I am aware of. But Pex does aid the TDD process (and other areas of software development), and as such reduces the cost while simultaneously increasing the quality of software development.