

Pex Sample: A Parameterized Model of the File System

Soonho Kong, Peli de Halleux and Nikolai Tillmann,
Research in Software Engineering,
Microsoft Research.

Preliminary Draft
Copyright Microsoft Corporation.

January 28, 2010

1 Introduction

Programs that interact with the file system are a classical challenge to the testable software. Using directly file system primitives such as the `System.IO.File` class makes the code less testable since it *hardcodes* a dependency on the state of the physical file system at the time tests are executed. A classic symptomatic example are test case that pass the first time they are run, then fail on subsequent run. The first run generated files which breaks subsequent runs.

To deal with these issues, programmers usually introduce a level of abstraction between the environment, i.e. the file system, and the code. In this sample, we took an abstraction of the file system that was defined in the *CodePex Client* project [1].

Unfortunately, even with a clear abstraction layer, the burden of the developer is still high. He has to understand the subtleties file system to craft a meaningful set of test cases. The file system is a complex system, whose complexity is underrated, which lead programmers to overlook obscure possible corner cases.

To address this, we will show how Pex can be used to define a parameterized model of the file system, `PFileSystem`. The sample will show how to implement a `CopyFiles` method that copies the files from one directory to another.

2 Sample sources

Follow these steps to unpack the sample solution and get ready for this tutorial.

- If haven't done so yet, unzip the `samples.zip` files. You can access it **Start|All Programs|Microsoft Pex|Samples|Samples.zip**.
- Open the `samples.all.sln` solution.

3 CopyFiles Method

The `CopyFiles` method copies all the file within the source directory into the target directory. Its behavior is just as the DOS command `copy sourcePath* targetPath`. Our first try uses the file system primitives `System.IO.Directory` and `System.IO.File`:

```
public static void CopyFiles(  
    string sourcePath,  
    string targetPath) {  
    var sources = Directory.GetFiles(sourcePath);  
    foreach (var source in sources) {  
        string target = Path.Combine(  
            targetPath,  
            Path.GetFileName(source));  
        // copy and do not overwrite  
        File.Copy(source, target, false);  
    }  
}
```

This `CopyFiles` is hard to test since it interacts with physical filesystem directly through classes under `System.IO` namespace such as `Directory`, `File`, and `Path`.

4 Introducing IFileSystem

To make the method testable, we need to abstract it from the file system. For example, by hiding the calls to `Directory` or `File` behind an interface.

The *CodePex Client* project [1] actually already did this work and published an interface `IFileSystem` that represents most of the operation that are commonly done on a file system. Its implementation using physical filesystem `FileSystem`. Both implementations are located in the `FileSystems` project.

We refactor the `CopyFiles` method to use the `IFileSystem` interface instead of the `Directory` or `File` classes.

```
public static void CopyFiles(  
    IFileSystem fs,  
    string sourcePath,  
    string targetPath) {  
    var sources = fs.GetFiles(sourcePath);  
    foreach (var source in sources) {  
        string target = fs.CombinePath(  
            targetPath,  
            fs.GetFileName(source));  
        fs.CopyFile(source, target, false);  
    }  
}
```

5 Test using PFileSystem

PFileSystem is a parameterized mock for the filesystem and it implements IFileSystem interface. Using this, we can use Pex to test CopyFiles method. Test method consists of the following steps.

1. We create PFileSystem instance, set source path and target path.
2. Copy files from the source path to target path
3. Assert that files in sourcePath also exist in targetPath.
4. Assert that each copy has the same contents as original.

Here is the code.

```
[PexMethod]
public void CopyFiles() {
    using (var fs = new PFileSystem()) {
        string sourcePath = @"\src";
        string targetPath = @"\tar";

        // CopyFiles
        SmallCopy.CopyFiles(fs, sourcePath, targetPath);

        // Assert
        string[] sources = fs.GetFiles(sourcePath);
        foreach (var source in sources)
        {
            string target = fs.CombinePath(
                targetPath,
                fs.GetFileName(source));

            // Assert 1 : files in sourcePath should exist in targetPath
            PexAssert.IsTrue(fs.FileExists(target));

            // Assert 2 : each copy has the same contents as original.
            byte[] sourceFileContent = fs.ReadAllBytes(source);
            byte[] targetFileContent = fs.ReadAllBytes(target);
            PexAssert.AreEqual(
                sourceFileContent,
                targetFileContent,
                (b1, b2) => b1 == b2);
        }
    }
}
```

After running the test with Pex, we have 10 passed tests and 8 failed tests. Passed tests show the files in source and target path after CopyFiles is executed. Note that the file names which Pex comes up with such as “;0;” and “\$ \$” are valid names even if it looks strange at first.

	files in source	files in target	Summary/Exception	Error Message
❌ 1			DirectoryNotFoundException	Directory "\src" does not exist.
❌ 2			DirectoryNotFoundException	Directory "\src" does not exist.
❌ 3			DirectoryNotFoundException	Directory "\tar" does not exist.
✅ 4	{}	{}		
❌ 5			DirectoryNotFoundException	Directory "\tar" does not exist.
❌ 6			UnauthorizedAccessException	Directory with the same target file name "\tar\===" already exists.
✅ 7	{\src\===}	{\tar\===}		
✅ 8	{\src\===}	{\tar\===}		
✅ 9	{\src\===}	{\tar\===}		
✅ 10	{\src\===}	{\tar\===}		
✅ 11	{\src\===}	{\tar\===}		
✅ 12	{\src\===}	{\tar\===, \tar\0}		
✅ 13	{\src\===}	{\tar\===, \tar\0000\000}		
✅ 14	{\src\===}	{\tar\===, \tar\0}		
❌ 15			IOException	The target file "\tar\===" already exists and overwrite flag is turned off.
✅ 16	{\src\===}	{\tar\===, \tar\::0;0}		
❌ 17			DirectoryNotFoundException	Directory "\tar" does not exist.
✅ 18	{\src\===}	{\tar\===, \tar\\$\$, \tar\=5}		
❌ 19			UnauthorizedAccessException	Directory with the same target file name "\tar\====," already exists.

Failed tests are categorized into three cases which we failed to consider in CopyFiles method.

- 1, 2, 3, 5, 17 : Source path("\src") or target path("\tar") does not exist in filesystem.
- 6, 19 : In target path, there is a directory whose name is the same as the file we attempt to copy("===", "====").
- 15 : The file we are copying already exist in target path.

With this information, we can modify CopyFiles to cover those cases. Here is the new CopyFilesImproved method. We added code to handle exceptional cases.

```
public static void CopyFilesImproved(
    IFileSystem fs,
    string sourcePath,
    string targetPath) {
    // check source and target path exist
    if (!fs.DirectoryExists(sourcePath))
        throw new DirectoryNotFoundException();
    if (!fs.DirectoryExists(targetPath))
        throw new DirectoryNotFoundException();

    var sources = fs.GetFiles(sourcePath);
    foreach (var source in sources) {
        string target = fs.CombinePath(
            targetPath,
            fs.GetFileName(source));
        // neither file nor directory should exist.
        if (fs.FileExists(target))
            throw new IOException();
        if (fs.DirectoryExists(target))
            throw new UnauthorizedAccessException();
        fs.CopyFile(source, target, false);
    }
}
```

Running the same test program with `CopyFilesImproved` method, we have 16 passed tests and no failed test.

	files in source	files in target	Summary/Exception	Error Message
✓ 1			DirectoryNotFoundException	CopyFilesImproved: Source path "\src" does not exists
✓ 2			DirectoryNotFoundException	CopyFilesImproved: Source path "\src" does not exists
✓ 3			DirectoryNotFoundException	CopyFilesImproved: Target path "\tar" does not exists
✓ 4	{}	{}		
✓ 5			UnauthorizedAccessException	CopyFilesImproved: Directory "\tar\=" already exists in target path "\tar"
✓ 6	{\src\ =}	{\tar\ =}		
✓ 7	{\src\ =}	{\tar\ =}		
✓ 8			IOException	CopyFilesImproved: File "\tar\=" already exists in target path "\tar"
✓ 9	{\src\ =}	{\tar\ =}		
✓ 10	{\src\ =}	{\tar\ =}		
✓ 11	{\src\ =}	{\tar\ =}		
✓ 12	{\src\ =}	{\tar\ =, \tar\ 0}		
✓ 13	{\src\ =}	{\tar\ =, \tar\ 0;}		
✓ 14	{\src\ =}	{\tar\ =, \tar\ S \$}		
✓ 15			UnauthorizedAccessException	CopyFilesImproved: Directory "\tar\=" already exists in target path "\tar"
✓ 16			UnauthorizedAccessException	CopyFilesImproved: Directory "\tar\;" already exists in target path "\tar"

References

[1] C. C. Team. Codeplex client, 2007. [accessed 9-October-2008].