# Use shims to isolate your app for unit testing

**Shim types**, one of the two key technologies utilized by the Microsoft Fakes Framework, are instrumental in isolating the components of your app during testing. They work by intercepting and diverting calls to specific methods, which you can then direct to custom code within your test. This feature enables you to manage the outcome of these methods, ensuring the results are consistent and predictable during each call, regardless of external conditions. This level of control streamlines the testing process and aids in achieving more reliable and accurate results.

Employ **shims** when you need to create a boundary between your code and assemblies that do not form part of your solution. When the aim is to isolate components of your solution from each other, the use of **stubs** is recommended.

## How to use shims

Let’s consider a method that throws an exception on January 1st of 2000:

```
// code under test
public static class Y2KChecker {

    public static void Check() {
        if (DateTime.Now == new DateTime(2000, 1, 1))
            throw new ApplicationException("y2kbug!");
    }
}
```

Testing this method is particularly problematic because the program depends on *DateTime.Now*, a method that depends on the computer's clock, an environment-dependent, non-deterministic method. Furthermore, the *DateTime.Now* is a static property so a stub type can’t be used here. This problem is symptomatic of the isolation issue in unit testing: programs that directly call into database APIs, communicate with web services, and so on are hard to unit test because their logic depends on the environment.

This is where shim types should be used. Shim types provide a mechanism to detour any .NET method to a user defined delegate. Shim types are code-generated by the Fakes generator, and they use delegates, which we call shim types, to specify the new method implementations.

The following test shows how to use the shim type, *ShimDateTime*, to provide a custom implementation of *DateTime.Now*:

```
//unit test code
// create a ShimsContext cleans up shims
```

```
using (ShimsContext.Create()
    // hook delegate to the shim method to redirect DateTime.Now
    // to return January 1st of 2000
    ShimDateTime.NowGet = () => new DateTime(2000, 1, 1);
    Y2KChecker.Check();
}
```

## How to use Shims

### Add Fakes Assemblies

1. In Solution Explorer, expand your unit test project's References.
2. Select the assembly that contains the classes definitions for which you want to create shims. For example, if you want to shim DateTime, select System.dll
3. On the shortcut menu, choose Add Fakes Assembly.

### Use ShimsContext

When using shim types in a unit test framework, you must wrap the test code in a *ShimsContext* to control the lifetime of your shims. If we didn't require this, your shims would last until the *AppDomain* shut down. The easiest way to create a *ShimsContext* is by using the static *Create()* method as shown in the following code:

```
//unit test code
[Test]
public void Y2kCheckerTest() {
  using(ShimsContext.Create()) {
    ...
  } // clear all shims
}
```

It is critical to properly dispose each shim context. As a rule of thumb, always call the *ShimsContext*. Create inside of a using statement to ensure proper clearing of the registered shims. For example, you might register a shim for a test method that replaces the *DateTime.Now* method with a delegate that always returns 1 January 2000. If you forget to clear the registered shim in the test method, the rest of the test run would always return 1 January 2000 as the *DateTime.Now* value. This might be surprising and confusing.

### Write a test with shims

In your test code, insert a detour for the method you want to fake. For example:

```
[TestClass]
public class TestClass1
{
        [TestMethod]
        public void TestCurrentYear()
        {
            int fixedYear = 2000;
            using (ShimsContext.Create())
            {
              // Arrange:
                // Detour DateTime.Now to return a fixed date:
                System.Fakes.ShimDateTime.NowGet =
                () =>
                { return new DateTime(fixedYear, 1, 1); };

                // Instantiate the component under test:
                var componentUnderTest = new MyComponent();

              // Act:
                int year = componentUnderTest.GetTheCurrentYear();

              // Assert:
                // This will always be true if the component is working:
                Assert.AreEqual(fixedYear, year);
            }
        }
}
```

Shim class names are made up by prefixing *Fakes.Shim* to the original type name.

Shims work by inserting detours into the code of the application under test. Wherever a call to the original method occurs, the *Fakes* system performs a detour, so that instead of calling the real method, your shim code is called.

Notice that detours are created and deleted at run time. You must always create a detour within the life of a *ShimsContext*. When it is disposed, any shims you created while it was active are removed. The best way to do this is inside a using statement.

You might see a build error stating that the Fakes namespace does not exist. This error sometimes appears when there are other compilation errors. Fix the other errors and it will vanish.