Interface Contracts Static Checking Example

Abstract

This example shows how to use contracts on interfaces.

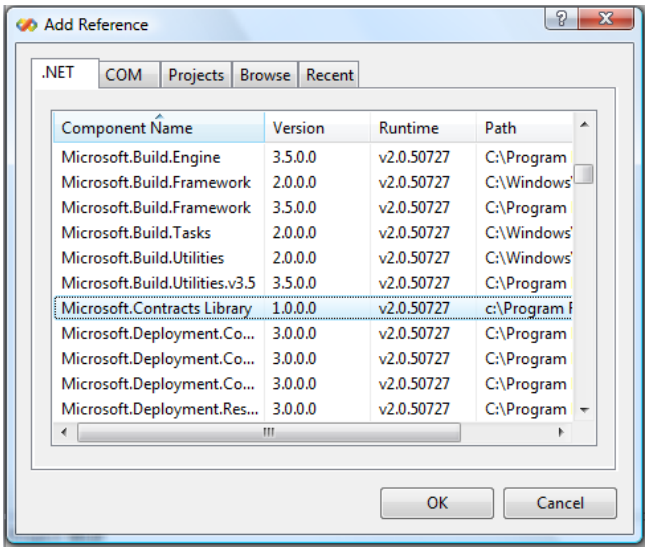
1 Adding the Contract Library Reference

If you are using Visual Studio 2008, or if you for some reason want to target a pre-v4 .NET runtime, then you need to:

- Change the target framework of the project.
- Manually add a reference to Microsoft.Contracts.dll

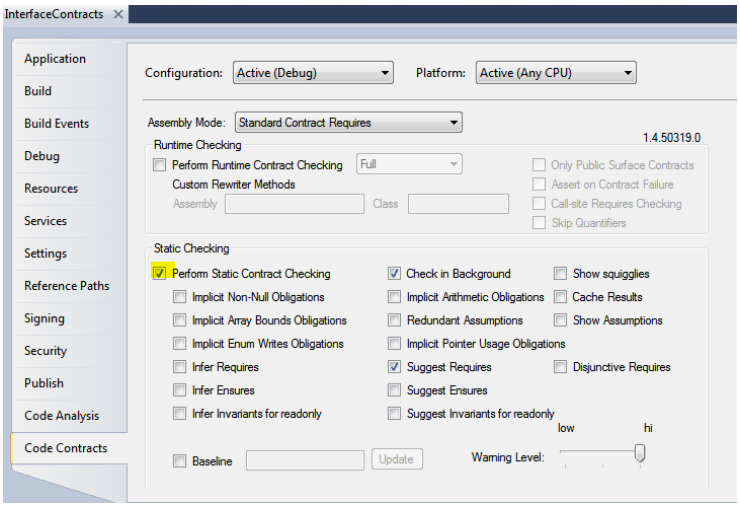
Otherwise, you may skip this section and go directly the next section!

To add the reference, open the InterfaceContracts solution and right-click on References in the InterfaceContracts project and select Add Reference. Find the Microsoft.Contracts library in the .NET tab as shown below and click OK.

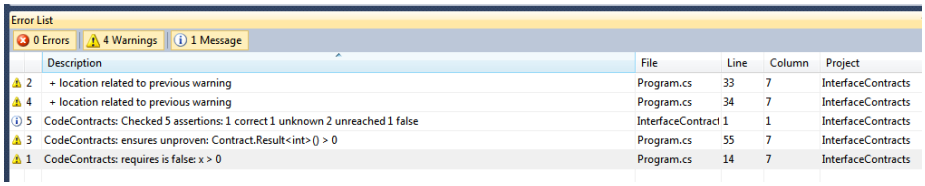


2 Sample Walkthrough

After adding the proper reference, go to the Properties of project InterfaceContracts, select the Code Contracts pane (at the bottom), and enable static checking by clicking on the checkbox as shown in this screenshot:



Then build the example. The static checker should warn about two problems:



The last problem in the list is in the call f1.Foo(0). Given that FooImplementation1 implements interface IFoo and IFoo has a contract written in the IFooContract class, the Foo method of FooImplementation1 automatically inherits these contracts, namely:

```
Contract.Requires(x > 0);
Contract.Ensures(Contract.Result<int>() > 0);
```

At the call site flagged by the checker, we are passing a value of 0, which is clearly not positive.

Note how contracts are associated with an interface: take a look at the interface declaration for IFoo. You see the attribute

```
[ContractClass(typeof(IFooContract))]
```

This attribute informs the contract tools that the contract for interface IFoo is to be found in the IFooContract dummy class. Now look at the IFooContract class: it similarly has an attribute stating what interface it annotates. This is for consistency and documentation.

```
[ContractClassFor(typeof(IFoo))]
```

The return value can be anything, or the methods can throw NotImplementedException.

Now take a look at the second error: it warns that the implementation of Foo in class FooImplementation2 is not conforming to the IFoo contracts. Namely, the method may return 0, if the argument is 1. Since the contract states that the return value should be positive, the contract is not satisfied. In contrast, FooImplementation1 properly implements this contract, as it returns the argument x, which is already known to be positive given the precondition.