.Configuring+the+Test+Scope+v10.4.0

In this section:

Configuring the Primary Scope

The test scope defines the set of files to analyze with dotTEST. You can specify the primary testing scope with the `-solution`, `-project`, and `-website` option. You can narrow down that scope by using the `-resource`, `-include`, and `-exclude` options. If you specify a solution or project as your primary scope, specify the solution or project configuration, as well as the target platform. Your command line may resemble the following:

```
dottestcli.exe -solution "C:\Devel\FooSolution\FooSolution.sln" -solutionConfig Debug -targetPlatform "Any CPU"
-config "builtin://Critical Rules" -report "C:\Report"
```

`-solution`: Specifies the path to the solution to be analyzed (you can specify more than one solution on the same command line).

`-solutionConfig`: Specifies the solution configuration.

`-targetPlatform`: Specifies the target platform.

`-config`: Specifies the test configuration to execute on the specified test scope.

If you are running analysis from your IDE, a source file that is open in the active editor has higher priority than resources defined with Solution Explorer and only this file will be analyzed.

Modifying the Scope with the `resource` option

You can narrow down the primary scope with the `-resource` option to specify one of the following:

- a path to a project in a solution
- a path to a directory of files in a project
- a path to a file

Paths must be relative to the solution. You can use Ant-style wildcards to specify the resource. I you want to provide absolute paths that match the file system, see Modifying the Scope with the `include` and `exclude` Options.

Examples:

Testing a Single Project in a Solution

```
dottestcli.exe -solution "C:\Devel\FooSolution\FooSolution.sln"
-resource "FooSolution/QuxProject"
-config "builtin://Demo" -report "C:\Report"
```

Testing a Single Directory of Files in a Project

```
dottestcli.exe -solution "C:\Devel\FooSolution\FooSolution.sln"
-resource "FooSolution/BarProject/QuxDirectory"
-config "builtin://Demo"
```

Testing a Single Source File

```
dottestcli.exe -solution "C:\Devel\FooSolution\FooSolution.sln"
-resource "FooSolution/BarProject/QuxDirectory/BazFile.cs"
-config "builtin://Demo"
```

Testing a Single Project Under a Solution Folder

```
dottestcli.exe -solution "C:\Devel\FooSolution\FooSolution.sln"
-resource "FooSolution/BarSolutionFolder/QuxProject"
-config "builtin://Demo" -report "C:\Report"
```

Testing a Single Source File When No Solution is Provided

Use the `-project` option only if the solution is not provided. Because the name of the solution is unknown, the solution path should start with `/`:

```
dottestcli.exe -project "C:\Devel\FooSolution\FooProject.csproj"
-resource "/FooProject/BarDirectory/QuxFile.cs"
-config "builtin://Demo" -report "C:\Report"
```

Modifying the Scope with the `include` and `exclude` Options

Use the `-include` and `-exclude` switches to apply additional filters to the scope.

- `-include` instructs dotTEST to test only the files that match the file system path; all other files are skipped.
- `-exclude` instructs dotTEST to test all files in the primary scope except for those that match the file system path.

You can use Ant-style wildcards to specify the resource. The following example shows how to exclude all files under directories `*.Tests`:

```
dottestcli.exe -solution "C:\Devel\FooSolution\FooSolution.sln"
-exclude "C:\Devel\FooSolution\*.Tests\**\*.*"
-config "builtin://Demo" -report "C:\Report"
```

You can specify a file system path to a list file (*.lst) to include or exclude files in bulk. Each item in the *.lst file is treated as a separate entry.

Specifying Additional Assemblies

Use the `-reference` switch to specify a path to additional assemblies needed to resolve dependencies of the analyzed projects. ANT-style wildcards and relative paths to the current working directory are accepted.

Examples

```
-reference C:\MySolution\ExternalAssemblies\*.dll
-reference C:\MySolution\ExternalAssemblies\*.exe
-reference C:\MySolution\ExternalAssemblies\**\*.dll
-reference C:\MySolution\ExternalAssemblies\**\*.dll
```

Use the `-reference` switch if you receive an "Unable to find reference assembly" message.