Integrating with Build Systems

Integrating with MSBuild

dotTEST ships with built-in support for integration with MSBuild to simplify running it in MSBuild build scripts environments. Integration with MSBuild scripts is achieved with the following custom MSBuild task:

```
Parasoft.Dottest.MSBuild.Tasks.dll
```

Use the following code in your MSBuild script after a task is deployed:

```
<Import Project="$(MSBuildExtensionsPath)\Parasoft\Parasoft.Dottest.targets"/>

        <Target Name="Demo">
                <Dottest
                        Configuration="builtin://Demo"
                        Solutions="C:\Devel\FooSolution\FooSolution.sln"
                         Report="C:\Devel\Report"
                         Out="C:\Devel\Out.txt" />
        </Target>
```

Target File

The target file must be imported in the MSBuild script. If you use deploy.exe, then you can the following import statement:

```
<Import Project="$(MSBuildExtensionsPath)\Parasoft\Parasoft.Dottest.targets"/>
```

Alternatively, you can import the target directly from the installation directory:

```
<Import
Project="[INSTALL_DIR]\integration\MSBuild\Parasoft.Dottest.targets\Parasoft.Dottest.targets
"/>
```

Targets can be directly imported from the installation directory when running multiple dotTESTinstallations on a single machine.

MSBuild Task

Use <Dottest> task in your build file to run dotTEST. The following arguments are supported:

- Configuration: Defines configuration used during analysis. See Configuring Test Configurations.
- Solutions: Defines solutions that are analyzed. See Running Static Analysis.
- Projects: Defines projects that are analyzed. See Running Static Analysis.
- Websites : Defines web sites that are analyzed. See Running Static Analysis.
- Resources: Defines resources that are analyzed. See Configuring the Test Scope.
- Report: Defines path to report. See Reviewing Results.
- Settings: Path to settings file. See Configuration Overview.
- NoBuild: Disables build of the tested solutions or projects.
- SolutionConfig: Solution build architecture. See Building Solutions and Projects.
- TargetPlatform: Solution build architecture. See Building Solutions and Projects.
- Out: Path where console output is saved.
- DottestPath: Path to dottestcli.exe file. Allows users to override the auto-detected dottestcli.exe path. This can be used to support multiple dotTEST installations on a single machine.

Integrating with NAnt

dotTEST ships with built-in support for integration with NAnt to simplify running it in NAnt build scripts environments. Use the following code in your NAnt script after the task is deployed to NAnt:

```
<target name="analyze">
        <dottest
                config="builtin://Demo"
                report="C:\Devel\Report"
                out="C:\Devel\Out.txt">
                <solutions>
                        <include name ="C:\Devel\FooSolution\FooSolution.sln">
                </solutions>
        </dottest>
</target>
```

Loading NAnt Task Library

Integration with NAnt scripts is achieved with the following custom NAnt task:

```
Parasoft.Dottest.NAnt.Tasks.dll.
```

This library must be in the same directory as NAnt.exe for the NAnt scripts to detect the dotTEST task. Alternatively, `<loadtasks>` can be used, which is useful for running multiple dotTEST installations on one machine

```
<project name="sampleProject" default="test">
        <target name="test">
                <loadtasks>
                        <fileset>
                                <include
name="[INSTALL_DIR]\integration\NAnt\Parasoft.Dottest.NAnt.Tasks.dll" />
                        </fileset>
                </loadtasks>
        </target>
</project>
```

Supported NAnt Task Arguments

- `config`: Defines configuration used during analysis. See Configuring Test Configurations.
- `solutions`: Defines solutions that are analyzed. List wildcards in ANT-style format separated by semicolons to analyze two or more solutions:

  ```
  <dottest config="builtin://Demo" solutions="C:\Devel\FooSolution\FooSolution.sln;C:\Devel\Bar\**\*.sln"
  />
  ```

  You can also nest `<include>` elements that point to ANT-style wildcards inside `<solutions>` elements:

  ```
  <dottest config="builtin://Demo" >
          <solutions>
                  <include name="C:\Devel\FooSolution\FooSolution.sln">
                  <include name="C:\Devel\Bar\**\*.sln">
          </solutions>
  </dottest>
  ```

  See Running Static Analysis.

- `projects`: Defines which projects are analyzed. List wildcards in ANT-style format separated by semicolons to analyze two or more projects:

  ```
  <dottest config="builtin://Demo" projects="C:\Devel\FooProjects\Qux\Qux.csproj;C:\Devel\BarProjects\**\*.
  csproj" />
  ```

  You can also nest `<include>` elements that point to ANT-style wildcards inside `<projects>` elements:

```
<dottest config="builtin://Demo" >
        <projects>
                <include name="C:\Devel\Foo\Qux\Qux.csproj">
                <include name="C:\Devel\Bar\**\*.csproj">
        </projects>
</dottest>
```

See Running Static Analysis.

- `websites` : Defines which web sites are analyzed. List wildcards in ANT-style format separated by semi-colons to analyze two or more websites:

```
<dottest config="builtin://Demo" websites="C:\Devel\Foo\WebSite;C:\Devel\Bar\*.WebSite" />
```

  You can also nest `<include>` elements that point to ANT-style wildcards inside `<websites>` element:

```
<dottest config="builtin://Demo" >
        <websites>
                <include name="C:\Foo\WebSite">
                <include name="C:\Bar\*.WebSite">
        </websites>
</dottest>
```

  See Running Static Analysis.

- `resources`: Defines which resources are analyzed. Separate two or more paths to resources with a semi-colon:

```
<dottest config="builtin://Demo" resources="Foo/Bar/Baz;Foo/Qux/Garply" >
```

  You can also nest `<res>` elements that point to paths inside `<resources>` elements:

```
 <dottest config="builtin://Demo">
        ...
        <resources>
                <res name="Foo/Bar/Baz" />
                <res name="Foo/Qux/Garply" />
        </resources>
</dottest>
```

  See Configuring the Test Scope.

- `report`: Defines path to the report:

```
<dottest
        config="builtin://Demo"
        report="C:\Foo\Report" >
        ...
</dottest>
```

  See Reviewing Results.

- `settings`: Defines the path to settings file:

```
<dottest
        config="builtin://Demo"
        settings="C:\Foo\settings.properties" >
</dottest>
```

  See Configuration Overview.

- `nobuild`: Disables build of the tested solutions or projects:

```
<dottest
        config="builtin://Demo"
        nobuild="true" >
        . . .
</dottest>
```

- `solutionConfig`: Solution build architecture. See Building Solutions and Projects.
- `targetPlatform`: Solution build architecture. See Building Solutions and Projects.
- Out: Path where console output is saved.
- `DottestPath: Path to dottestcli.exe` file. Allows users to override the auto-detected `dottestcli.exe` path. This can be used to support multiple dotTEST installations on one machine.