Workspaces, Projects, and Test Files

Parasoft SOAtest applies the concept of a *workspace* as it builds on top of the popular Eclipse IDE. This is equivalent to a “solution” in Microsoft Visual Studio. The workspace corresponds to a directory on the local file system. The location of this directory generally is not important. SOAtest defaults to a certain location on the local machine, but each user/team can change it if they wish.

A workspace can contain multiple *projects*, each of which correlates to a directory inside the workspace on the local machine. The project can contain multiple .tst files along with any related files and artifacts such as data source Excel spreadsheets, keystores, etc. These artifacts may reside directly under the project, or be organized under subfolders within a project.

The project level is typically the most suitable level for team sharing. As a result, it’s important to consider the layout for the projects’ .tst files and the file resources that they depend on.

Project Layout Patterns

Developer-Focused Teams

If developers are the primary users of SOAtest in your organization, you may benefit from SOAtest being packaged as an Eclipse IDE plugin. This enables developers to leverage SOAtest’s capabilities directly from within the IDE where they work on source projects.

In this case, SOAtest-related files can be kept under the source projects so they can be maintained along with the source code. A “tests” directory within a source code project can contain all the .tst files and their dependencies.

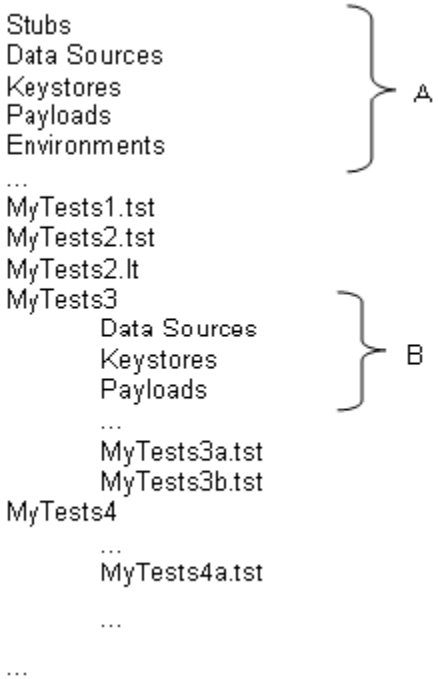
Another advantage of this approach is that when any custom Java extensions are developed for SOAtest (scripts, etc.) or when messaging tests require vendor jar files (for JMS, MQ, etc.), then SOAtest’s classpath system properties can be configured to inherit directly from the Java project’s classpath—so you don’t need to manage such jar files separately or duplicate them for the purpose of testing.

QA-Focused Teams

If your SOAtest users are primarily Quality Assurance engineers, who typically do not access source code, you probably want to have tests maintained under projects that are created specifically for SOAtest purposes.

SOAtest Test File Structure

Here is one possible way to organize SOAtest .tst files and related resources:



Shared resources can be stored in folders in the A category. These tend to be data source files, environment configuration files (more on that later), or keystores that are widely applicable to many .tst files. For example, this category might include data sources with authentication data that is broadly applicable, database connection configuration files, and so on.

Folders in the B category have resources that are specific to that collection of tests. For example, MyTests3 could be the directory where Web UI tests (with their own dependencies and data sets) are stored, while MyTests4 is where Web service tests are kept. Note that folders in this category may sometimes have the same name as the folders in the A category.

When working within a .tst file, you can reference resources that are stored in any level within the project (up or down the hierarchy) while maintaining relative path referential integrity.

The **Persist as Relative Path** option can be used to create a path that is relative to the current configuration file (rather than a full absolute path). When this option is enabled, Eclipse variables will be inserted as needed—for example, {project_loc:MyProject}/ReferencedTest.tst. Using relative paths makes it easier to share tests across multiple machines.