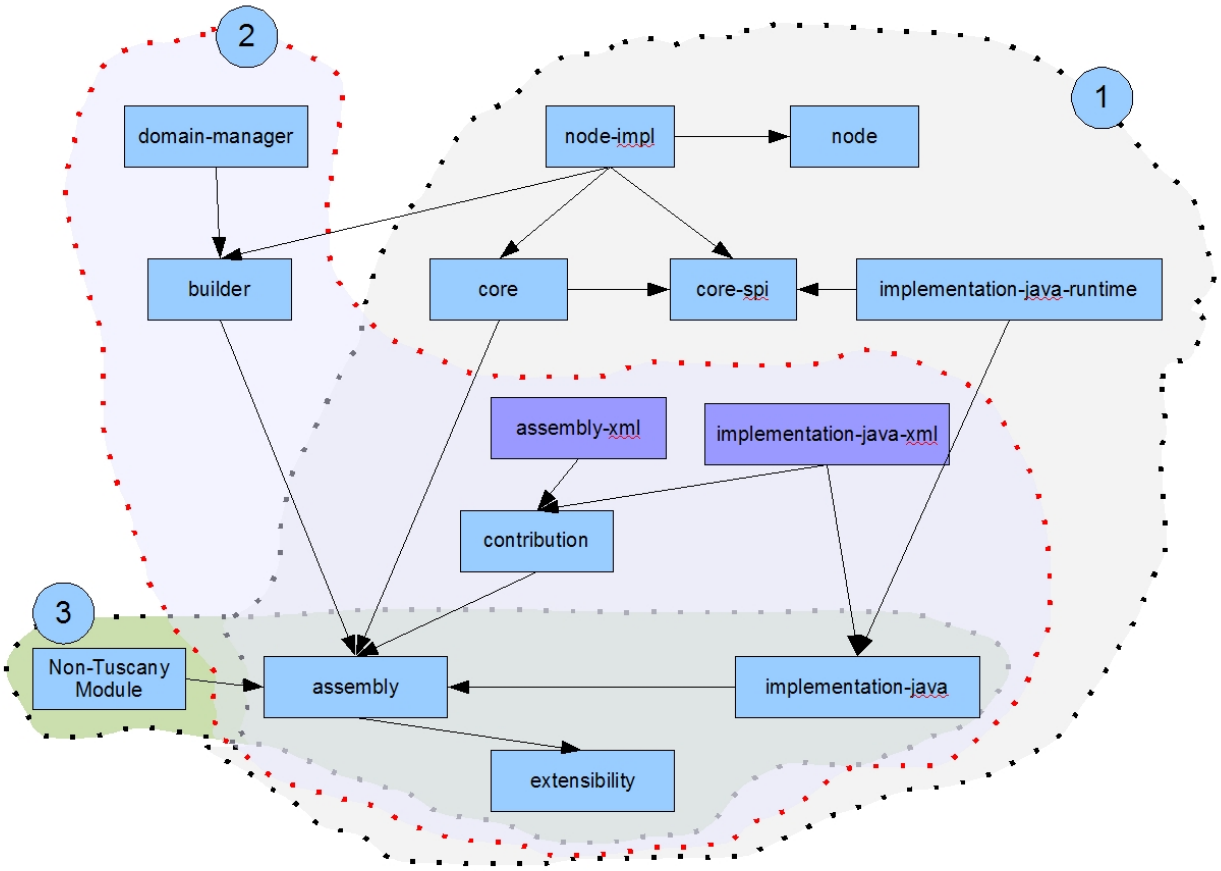
Tuscany modularity, composability and usability

Tuscany modularity, composability and usability



We have been discussing Tuscany modularity, composability and usability over and over. There seems to be a lot of confusions as we use different terms, mix different goals and have different opinions. I would like to bring all the relevant items to the table and you can add your own. Let's try to agree on the following items one by one. After that, we can come up a plan and execute it against the current structure in a batch to avoid disruptive changes.

1) Terminology and target audiences

1.1 Module: A maven module in the build, a project in Eclipse, or a jar (or OSGi bundle) in the distribution. This is visible to Tuscany developers and those who wants to extend Tuscany or embed Tuscany.

1.2 Feature (or Profile): A mechanism to describe and package a set of related modules that can be used together for an offering. This is visible to Tuscany developers and those who wants to extend Tuscany or embed Tuscany.

1.3 Distribution: A downloadable archive of an offering. Tuscany can provide all-in-one package and other smaller packages such Web Services, Web 20, Enterprise Java etc. This is visible to Tuscany end users. It should be also possible that a user unzips another offering into the existing Tuscany directory.

2) Key principles of Tuscany architecture

Tuscany is a composable platform that provides a set of modules and services that can be combined together to build an offering. Not all modules are required for all offerings. Tuscany is organized as a code base that we will use to build a variety of offerings. The Tuscany platform itself is designed to provide the common modules needed by many offerings and to provide them in a way that is reusable and generic. A specific offering (such as a Tuscany Node, Tuscany SCA Domain Manager or SCA Tool) would select a set of Tuscany modules it needs and then add a set of its own modules to provide the offering specific content. Applications (as defined by the offering) would then run on this collection of modules. Applications (and end user administrators) would likely also have some choice over which sets of modules are in use in a given environment.

The promises are illustrated in diagram at 1. It shows Tuscany or other products can compose offering from the modules in different ways.

1 http://cwiki.apache.org/confluence/display/TUSCANYWIKI/Tuscany+modularity%2C+composability+and+usability

3) How to justify if two functions should be in the same module or separate module?

3.1 Are the two functions (A and B) can be used independently?

3.1.1 If A depends on B at compile time, then try to find out if another function C (inside or outside Tuscany) can use B without A. If such a function C exists, then A and B should be separate modules.
3.1.2 If A depends on B at runtime, there is a good opportunity that B can be replaced by another implementation (like the stax-api and woodstox). We should have A and B in separate modules.
3.1.3 If A doesn't depend on B, we should have A and B in separate modules unless they are both trivial and it won't add dependencies or footprint to have them in the same module.

It is very important to distinguish between "compile" and "runtime" dependencies.

3.2 Do the two functions belong to different layers?

- Two functions should be in separate modules if they belong to different layers
- If they belong to the same layer, then use the rules in 3.1.

3.3 Good ways to keep modularity

- Don't use "compile" dependency unless it is required for compilation
- Use Extension Point/Extension pattern via interfaces for cross module calls if necessary to avoid hard dependencies

4) How to describe a group of related modules to provide functional offerings?

- Maven pom and assembly descriptors to describe the dependencies and packaging
- Maven profiles to enable the build for each offering

5) How to distribute a set pre-defined offerings from Tuscany?

- For each offering, we use the maven assembly plugin to produce an archive.

Thanks,
Raymond