JMeterArchitecturalOverview

This page describes the main JMeter pieces: the GUI, the test tree (aka Test Plan), the Engine,... and how they relate to each other.

PLEASE NOTE THIS PARTICULAR PART OF JMETER'S ARCHITECTURE IS CURRENTLY BEING HEAVILY MODIFIED BY OLIVER ROSSMUELLER. (2003-01-08)- I would like to know more about exactly what you are doing before you commit. As I said in email, I don't think what you are doing should be all that complicated. - MikeStover

- It did not look that complicated when I started but I could not resist to change and clean up stuff while I was walking through the code. It is like a chain reaction: at first I just wanted to implement cut/copy/paste and now I'm in the middle of nowhere. Don't get me wrong, I'm not lost in the dark, I know where I am, where I have to go and what I have to do, but it will take some time. To let you know where I am at the moment I added a section below where I will describe what I've done so far and what I will do the next time. -- OliverRossmueller
- What will you do if the rest of the developers like some of the changes, but not others? Having it all wrapped up in one great lump in a separate branch is not going to work well. If I were you, I'd stop and start the discussion of these things, because it may turn out your wasting your time. I can tell you now there's several things I strongly disagree with in your list below. -MikeStover
- Refactoring = changing the the intern organisation without changing the behavior! Why do you add features? -- JörgWeinmann

Each node in the test tree is currently represented by a JPanel -- shown on the right side of the interface when you select that node.

These JPanel objects are created as you build the test plan (or all in a shot if you load a test plan from scratch) -- this causes performance concerns (both in terms of time and memory).

Those JPanels implement the JMeterGUIComponent interface. The most relevant method defined by this interface is the createTestElement() method -- which will create a Test Element from the values held by the GUI.

The Engine takes that tree and compiles and runs it. See JMeterTestExecution for more info.

Open questions:

- Who invokes createTestElement()? The engine? Also when the tree needs to be saved to disk? - **Answer:** The Actions control this behavior. Start.java queries the JMeterTreeModel for the TestTree, and then converts it into TestElements via an inherited method (inherited because other actions use it to). See AbstractAction.java.
- Are test elements also created and used to populate the GUI when a test plan is loaded from disk? - **Answer:** Yes, the process goes in reverse here. GUI objects are populated with information from the TestElements via their configure(TestElement) method.

What's going on on the refactorings_branch

- changed the way the tree in the GUI is working
  - the tree operates on the test elements directly
  - there is one single instance per GUI class which is reused to display/edit all test elements of the kind the GUI is responsible for
  - all existing GUIs were refactored/rewritten for a consistent (and hopefully better) look
    - Advantage: reduced memory footprint because of less GUI components; less complicated operations on the test elements as they are always up to date (no traversing of GUI component tree necessary to get the test element tree)
    - Disadvantage: implementing the GUIs is a little bit more complicated now as all changes in the GUI have to be propagated to the corresponding test element immediately
- searching the complete classpath for GUI classes took quite some time and slowed down the start of JMeter - I introduced the concept of plugins to get around this
  - a plugin directly registers all it's test element classes, GUI classes, icons, message resources
  - core, components, http, ftp, jdbc, java are plugins now
    - Advantage: JMeter starts faster
    - Disadvantage: implementing new { { { elements/GUIs } } } is a little bit more complicated because of the need to create a plugin for JMeter to find your new classes, it is not possible to just add them to the classpath any more
- map in the test elements holding the properties and children was removed and replaced by instance variables and the use of special property objects
  - Advantage: it is clear on the first sight which properties a test element has because of using instance variables
  - (Potential) advantage: handling of user variables should be easier because value substitution can be done in the property object and is transparent to the user of the property.
- mapping from old to new property names necessary to load .jmx files -> proof of concept exists, has to be implemented for all test elements
- internal reorganisation makes it almost impossible to write .jmx files that are compatible with JMeter 1.8 -> introduced a new file format, save is implemented as proof of concept, load has still to be done
- finally I added some new features:
  - splash screen
  - toolbar for core operations; possible to extend with buttons to create test elements
  - tabbed main panel to edit multiple test plans in parallel