JMeterGuiTestElementSeparation

JMeter GUI Classes and TestElements

Problem (Oliver, please correct) - JMeter TestElements are currently saved with a String that provides the name of the GUI class associated with it. This is what allows JMeter to provide the correct GUI for a TestElement. This creates a problem for anyone who wants to create test scripts outside of JMeter, because only the GUI components know which TestElement they work for.

Solution - Create an external mapping that associates each TestElement class with a GUI class. This mapping would be available to anyone to read and use for the purpose of making JMeter test scripts. Also part of this solution is changing the way TestElements are created in JMeter's client app.

- Need specific steps and description of how TestElement creation will be handled

---

Problems with the solution

1. Creating a 1:1 mapping of test elements to gui objects will end the current practice of allowing a single TestElement class to be served by multiple GUI classes. This will require that every config element be given it's own, essentially content-less class. It's inconvenient to have to make a new class just for NameSpace reasons.
2. It prevents a developer from providing an alternate GUI view of a TestElement. It would be a nice touch to have two versions of the HTTPSampler view - one that shows all the options, and one quick-and-dirty view that just has a URL field.
3. When writing a new component for JMeter, a new task has been added to the developer's list of tasks that need doing: update the config file that provides the mappings.

Solutions for the problems

1. Regarding probs 1,2: forget about allowing multiple GUI's serve the same TestElement. It's a minor inconvenience to have to create new empty classes for each new GUI.
2. If a tool was written that searched through JMeter's jars and compiled an XML document that specified all the necessary information, it would greatly reduce the work a developer would have to do.
3. Does the mapping have to be reversible? Surely all an external test developer needs is to provide a valid GUI for a given test element? In which case, all that needs to be done is to provide a *preferred* GUI class for each test element.sebb