FutureReleases

Discussion area for what enhancements to put into future releases.

If you are not a JMeter committer, feel free to add comments (with initials please), but please don't reorganise the release plan.

Possible enhancements

This section is for listing possible enhancements. Describe them briefly. If necessary add bug ids (or links) to clarify.

- JDBC connection pooling - possible enhancements
  - use Apache DBCP instead of Excalibur?
- JDBC sampler - use different cache sizes for connection and statement?
  - How do these really work?
  - Should the connection cache be a ThreadLocal item?
  - Need to add debugging to show when cache items are created and destroyed - difficult for pooled resources
- XPath Extractor - add user-declared namespaces
- Rework HTTP GUI to make it easier to use and extend
- Add HTTP methods dropdown to "SOAP/XML-RPC sampler", bugzilla 42637
- Remove unnecessary code duplication in "SOAP/XML-RPC sampler", rather use code inherited from HTTPSampler2. This is almost done, not committed to svn (Alf Hogemark)
  - Add unit tests for SOAP/XML-RPC sampler
  - Change name exposed to use to "SOAP/XML-RPC/REST sampler"
- Make it possible to send file content for HTTP GET, PUT and DELETE using "HTTP Sampler" and "HTTP Sampler 2", and therefore also "SOAP /XML-RPC sampler" <= some of this ha been done
- Make Axis2 / XFire / CFX Sampler, and possible retire existing "Webservice(SOAP) sampler".
  - A bit unsure about this, not sure what benefit is over "SOAP/XML-RPC Sampler".
  - Will we really be testing Axis / XFire client side peformance here ?
  - Or could we make a nice GUI, so it is easier to test web services ?
- Make command line option available, to easily performance test one url
  - This is meant as a replacement for the "ab" command, and alternative to "faban"
  - I think it would broaden the user of JMeter a lot
  - I will look into making a "ab result listener", which presents the same statistics as "ab" initially (Alf Hogemark)
  - One possible implementation is
    - Add a simple jmeter test plan, with threadgroup, http sampler, and listener, using property values to control behavior to the bin /system_testfiles directory
    - Add shell scripts, for example jmeter_test_http, which just calls the standard jmeter script, with additional "-J" arguments, for example "-Jiterations=10"
    - Need a way of getting the listener output to the console, need some investigation
      - on Windows one can use CON as the file name; /dev/tty for unix - but output is not formatted nicely
      - Summariser can already output to console - that may be enough
- Restructure HTTP Sampler / Settings GUI. bugzilla 41917. A big job, but is becoming necessary, if we are adding more options to the HTTP Sampler
- Listeners:
  - make distinction between open file for read and write.
    - at present the browse button does both, which is confusing (given the heading) <= this has been changed
    - a file that has just been read will then be used for writing if a test is run - is that sensible, or should the file name be cleared after it has been read?
  - Graph Transactions Per Second (TPS) like the graphing of Response Time.
- Add a "Preview" button to "CSV Data Set Config", which would use the file name and parameter specification, and bring up a dialog or something to show an example of the variables available and their values.
  - Useful for easily checking that you have specified the columns and variables correctly
  - Could this limit the number of emails on the jmeter-user list asking how "CSV Data Set" works
- Improve documentation
  - Currently, there are quite a lot of question duplication on jmeter-user mailing list, would be nice to reduce that number
  - Does people really read the documentation ? Judging by posts on jmeter-user mailing list, I sometime have my doubts.
  - Would be excellent to have documented steps for "view frontpage, log in, view a page, log out" scenario for the following frameworks
    - Struts
    - Spring
    - Spring Webflow
    - .net applications
    - others (please suggest)
    - I think that could reduce number of mails to jmeter-user mailing list
    - the above could start out as Wiki pages
    - it would be useful to provide sample JMX files as well
- Additional sample JMX files for the various scenarios that are FAQed
  - Perhaps a giant example showing lots of different scripts?
- Support for uploading multiple files in HTTPSampler / HTTPSampler2. I think it the one main missing functionality in HTTPSampler that browser provide, and JMeter not.
  - this is bugzilla 19128
  - I think this is dependent on bugzilla 41917, mentioned above
- Improve support for testing web service. This overlaps with some of the points above.
- Make a "HTTP Caching Manager", this is bugzilla 28502.
  - It is useful for performance testing
  - Would also be useful for functional testing, to see if pages are "cacheable".
- Make a "HTTP Proxy Manager", where the user can specify what proxy settings to use

    - Then use would not have to edit jmeter.properties file or command line arguments to use a proxy
    - We could get rid of the "HTTP Proxy" settings in the "Webservice(SOAP) sampler"
    - Could perhaps just extend HTTP Defaults for this?
    - N.B. HTTP (Java) relies on system properties to define proxy items, so it is impossible to have more than one setting.
- Make new, nice?, icons for the GUI elements which currently do not have a unique icon. For example the post and preprocessors. <= The Pre and Post-Processors now have different icons (corner fold is in a different place)
    - Would make it a lot easier to see what is what in the tree
- Extend CSV Data Set to read via JDBC
- Assertion Results (or similar) could be used to attach errors to samples - e.g. if a Post-Processor failed, the error could be shown there instead of in the log. Or just add string array for storing such errors?
- Module Controller GUI
    - should it omit leading path name elements as at present?
    - it would be nice if it could check for ambiguous controller names:
        - report error if ambig. name selected? OR
        - flag ambiguous names in the drop-down list?
- Java Sampler to invoke arbitrary Java method. Would need:
    - classname, method name, params (and types) , classpath
    - this can be done currently with BeanShell, but it would be simpler to have a dedicated GUI.
- Remote Mode selection: might be useful to be able to specify the remote mode for each listener. This would allow the use of immediate mode for a few listeners and statistical mode for most of traffic.
- If Controller - allow functions and variables as alternative to Javascript
- Rework architecture to be able to use a pool of threads instead of thread or use Reactor Pattern
- Introduce HTTP Client async feature
- Allow parallel HTTP requests for one simulated user
- Support Websocket and SPDY2 (HTTP/2)
- Fix known Bugs in REDO/UNDO feature
- Add a StreamProcessor to look at/modify the data of Samplers while they are retrieved/sent. Could be used to verify that a large video stream is valid while not consuming a lot of memory.
- Extend the functionality of the imap sampler.

Non-functional improvements

This section is for non-functional changes, e.g. code re-organisation etc.

- Currently, Apache Avalon seems to be used for logging. Avalon seems to have stalled, should we move to log4j ?
    - Would we gain functionality by moving to log4j ?
    - Alternatively, move to Commons Logging (as used by HttpClient currently)
    - Or perhaps SL4J - http://www.slf4j.org/
    - What are the risks with continuing to use Avalon, if Avalon is not maintained anymore ?

    - Remark: [HttpComponents] project is considering migrating [HttpClient] 4.0 branch off Commons Logging [this was decided against: sebb] to another logging framework as Commons Logging currently appears unmaintained [not so: sebb] and the prospects of its further development remain unclear. [There has just been a new release: sebb] It may be worthwhile to ensure both projects use the same logging framework / compatible logging frameworks. [Agreed: sebb]

- Reorganise documentation
    - component_reference is getting much too big. This will require changes to the help system.
    - sort functions and component ref into more logical order (currently chronological)
    - perhaps use separate XML files for each item, included by main pages ?
    - Could we then add a "Help" button to each GUI element, which would bring up the correct help in the browser ?
    - or extend the existing help menu to load just the individual page.
    - If there are combined and individual help pages, there would probably need to be two copies. Maybe simplest just to split component reference by element type; keep current page as an index into the subsections
    - Help could perhaps be extended to allow loading of linked pages (but one would probably not want to allow external links to be loaded). There is some code in View Tree that might help here.
- Re-write ClassFinder:
    - needs general tidyup / javadoc
    - cache results - same classes may be requested multiple times
- TestBean:
    - prepare is probably called too often; can it be done once per test?
    - need some more GUI types - eg. table
    - would be nice to be able to enable/disable fields depending on what else is selected - e.g. JDBC parameters only needed for prepared statements
    - ensure drop-down list size big enough for all entries (within limits!)
- move from Bugzilla to JIRA? More flexible, (but attachments a bit more awkward at present?)
- JMS GUIs should be loadable without needing JMS jars (needs an extra level of indirection, as is done by JMS Publisher) or as below.
- How to handle Gui elements that depend on optional jars:
    - should these be displayable, even though the jars are missing? Convenient for creating and viewing test pans, but not so useful at run-time - should the test plan be allowed to run?
    - or should they be omitted as at present? - this is confusing at build time.
    - or perhaps generate a dummy entry in the list, with a message to say the jar is missing? his would be tricky, as the class is needed to retrieve the name and the menu category. Perhaps the way to do it is to handle it in the GUI by catching the errors, and changing the name or screen comment? May be tedious to do.
- Sort test tree according to JMeter processing order? This should probably be a separate action, as it would be confusing for the tree to change as it was editted! (the menu drop-downs are now in the correct order)
- GUI code refactor
    - there are various table implementations, could they be combined?
    - perhaps the table models could also be combined?

- Add SVN revision number to version? (already added to Manifests)
- Consider migration to Maven2 as a build tool for JMeter. This should help simply dependency management and facilitate the use of JMeter for automatic load testing / integration into tools like Continuum
    - Maven 1 was tried a couple of years ago, and seemed incompatible with the JMeter directory layout and multiple jars; hopefully Maven 2 is more flexible.
- Consider renaming ThreadGroup as JMeterThreadGroup - or UserGroup ? - to avoid confusion with java.lang.ThreadGroup
- Documentation refers to threads and/or users in different places; replace by users - or users(threads) - everywhere?
- Test elements need access to ThreadGroup and TestPlan for co-ordinating counts etc per-group and per-plan. Not much distinction is currently made between per-group and per-plan.