BELLEROFONTE

HOW TO EXTEND WITH NEW TESTS IN 15 STEPS
01/08/2003 - ver. 0.86

---

1. Anzitutto si deve **individuare la libreria adatta** alla realizzazione del nuovo test: o si sfrutta solo la libreria di default (HTTPUnit) o se ne deve importare una nuova (o più) in aggiunta. Se la libreria scelta non è compresa tra quelle già utilizzate nei precedenti test allora
   - copiarne il relativo file jar nella directory /jars
   - editare il file *manifest.mf* in /lib aggiungendo nella riga "Class-Path" la scritta "../jars/" + <nome Jar aggiunto>
   - aggiungere il nuovo archivio jar nel Path dell'editor usato per scrivere il nuovo test

2. una volta scelta la libreria che offre le giuste funzionalità, aprire il file ***Anonymus.java*** che si trova in /src e salvare subito con altro nome il nuovo test. Anonymus è lo "scheletro" di ogni nuovo test e dovrebbe rimanere tale

3. **rinominare la nuova classe ed il suo costruttore** con il nome del nuovo test. E' importante che ogni classe test resti estensione di *Single_test* e implementi l'interfaccia *Serializable*

4. nel nuovo test creato da *Anonymus* sono individuabili 5 **aree personalizzabili**. Si deve procedere al loro corretto riempimento evitando di modificare il restante codice, che serve a Bellerofonte per gestire i test singoli a prescindere dalla loro funzione. Qualora la funzione di un'area non risultasse chiara è consigliabile l'ispezione del listato di qualche altro Single test già implementato. Si ricorda che è sempre buona norma commentare adeguatamente ciò che si inserisce all'interno delle aree modificabili

5. la **prima area** concerne **l'importazione dei pacchetti** contenenti le librerie che servono alla realizzazione del nuovo test. Aggiungere dunque il nome di pacchetto di ogni elemento utile

6. la **seconda area** prevede l'inserimento dei **campi** del test. I campi del test non sono altro che le variabili stringa contenenti gli eventuali valori ricevuti in input dal file ".txt" o ".tst" di definizione del test. Ad esempio, se in *test.txt* c'è una riga contente `CheckTitle|mio Titolo|`, "mio Titolo" è il parametro di input passato al test CheckTitle che sarà in seguito assegnato ad un apposito campo. Non tutti i test hanno bisogno di campi

7. la **terza area** è contenuta nel metodo *setUp*. In quest'area **si inizializzano tutti i campi** eventualmente specificati con i relativi parametri in input. Si ricorda che i parametri sono contenuti nel vector *parameters* **a partire dalla locazione 1**

8. la **quarta area** è nel metodo *execute* e rappresenta ciò che viene visualizzato nel report come **risultato atteso**, subito dopo l'intestazione (identica per ogni test)

9. la **quinta area** costituisce **il test vero e proprio**, cioè contiene il codice necessario alla realizzazione del test:

- se si vuol utilizzare la libreria di default è consigliato andare a leggere dal vector *pageData* gli elementi tipici della conversazione web HTTPUnit*: WebConversation, WebResponse, ClientProperties*. A questi elementi possono fare riferimento tutti i test che importano la libreria di default ed in generale il vector *pageData* è usarto per scambiarsi informazioni e aggiornare lo stato della pagina attualmente analizzata. Per ottenere dentro un test questi elementi, si deve effettuare un *cast* come nel seguente esempio:
  `WebConversation resp = (WebConversation)pageData.elementAt(0);`
  Analogamente può avvenire per *WebResponse* e *ClientProperties* sapendo che si trovano rispettivamente nella locazione numero 1 e 2 di *pageData*;
- se in aggiunta o in sostituzione della libreria di default si vuol usare un'altra libreria Java, si può procedere ad una lettura della pagina secondo i suoi oggetti. Al termine del test è possibile prevedere il salvataggio di alcuni oggetti in coda al vector *pageData*. Per rendere visibili anche ad altri test le informazioni aggiuntive (ed in generale ogni volta che un test modifica la pagina web), *pageData* deve essere salvato nella Web_page con il metodo:
  `page.setCurrent(pageData)`.

Di solito ogni test prevede in quest'area anche la gestione di eccezioni particolari. Ogni eccezione dovrebbe comparire nel report con un idoneo messaggio di segnalazione e dovrebbe assegnare la stringa "Error" alla variabile *_finalResult*. Infine, nei casi in cui non ci sono stati errori generali, ogni test dovrebbe annotare nel report se è terminato con successo o con insuccesso altrimenti il suo risultato non verrà registrato. Il codice necessario a queste operazioni si trova, commentato, all'interno di quest'area

10. quando il test presenta caratteristiche di **complessità elevata** si consiglia di realizzare dei **metodi privati** interni e accessori, da richiamare nelle aree descritte. Come esempio si può visualizzare il metodo *visitDep* del test *ReachAll*. Seguendo la stessa filosofia, se per realizzare il test si devono implementare svariate **classi accessorie**, per evitare di mescolarle alle altre classi si consiglia di includerle in un'apposita **libreria** da inserire assieme agli altri jar. Anche in questo caso si può visionare la classe ReachAll.java, la quale importa la libreria accessoria *claudiosoft.struct.btree*

11. a questo punto, se il codice Java del test compila correttamente, si procede **all'aggiunta del nuovo file al progetto** orbilio.bellerofonte. Ricompilato l'intero progetto si deve verificare che l'output della compilazione (ovvero tutti i file ".class", compreso quello del nuovo test) sia stato correttamente inserito in /orbilio/webTest/bellerofonte. E' in quella directory infatti che Bellerofonte andrà a cercare la definizione del nuovo test invocato

12. si procede poi ad **editare il file "*TestList.txt*"** contenuto in /webtestfiles/program. Esso contiene un elenco dei nomi di tutti i Single test per i quali esiste una definizione. Per aggiornarlo basta aggiungere in fondo alla lista dei test attualmente riconosciuti, il nome del nuovo test creato. E' importante che sia rispettata l'esatta sequenza di maiuscole e minuscole

13. **editare il file *TestSynopsis.txt*** in /webTestFiles/program aggiungendo l'esatta segnatura e una sintetica descrizione del nuovo test. Per permettere una maggior comprensione di come il test interagisce con gli altri si indicano inoltre: le librerie usate, gli eventuali valori di ritorno al Macro test (rari: implicano una modifica a *Macro_test* stesso), gli eventuali valori modificati o inseriti nella *Web_page* tramite il vector *pageData*. Esempio di test che ritorna valori è *FollowLinks* mentre uno che modifica i valori di *Web_page* è *SubmitForm.*

E' consigliato aggiornare anche l'equivalente documento contenuto in /various, per un layout più idoneo alla stampa

14. **eseguire lo script batch dos makeJar.bat** in /bin, il quale aggiunge al file *Bellerofonte.jar* in /lib il file ".class" del nuovo test. In caso di utilizzo di nuove librerie provvede inoltre ad aggiornare la variabile di sistema *Classpath*

15. dopo aver allestito gli appositi file di test e di opzioni così come illustrato nei documenti contenuti in /various, **eseguire nuovamente belJar.bat** in /bin.

Se la procedura è stata eseguita attentamente il nuovo test dovrebbe essere adesso integrato a Bellerofonte.

Infine:

- si consiglia di evitare librerie incompatibili con quelle già integrate;
- se si costruiscono nuovi set di test che presuppongono l'utilizzo di particolari locazioni del vector "pageData" per la lettura e la scrittura di informazioni, si consiglia di indicare questa particolarità nel TestSynopsis.txt. Questo per evitare che si tenti di utilizzare una sequenza di test che presuppongono degli input e degli output non previsti.