Logging in Geoweb: best practices

In Geoweb, il sistema di log lato server è basato su **log.apache.log4j.** Una certa quantità di log è generata in automatico da componenti del framework come Spring (MVC), MyBatis e workflow (Activiti). Altri log sono in genere creati in concomitanza all'esecuzione dei service/controller di Geoweb, e generalmente sono usati per loggare parametri in ingresso, risultati di elaborazioni, eventuali eccezioni… Per dare risalto a tali log, in modo da poter essere più facilmente indivuati e letti dagli utenti sia sviluppatori che configuratori, sono state create una serie di funzionalità di base. Queste funzionalità hanno anche lo scopo di uniformare in tutto il framework le modalità di presentazione dei log all'utente. La centralizzazione di ciò permette in qualsiasi momento di effettuare modifiche e vederle applicate a tutto Geoweb.

La classe ***GwLogService*** (presente nel package *com.geowebframework.webclient.service) GwLogService* è disponibile ovunque così:

```
    @Autowired
    private GwLogService gwLogService;
```

Esso espone il metodo ***.logMessage(GwLogMessage gwLogMessage)*** che prende in ingresso istanze della classe ***GwLogMessage*** (presente nel package *com.geowebframework.transfer.objects.log*). Questa possiede tutto un set di variabili private per memorizzare, tra l'altro:

- tipologia di messaggio: DEBUG, INFO (default), WARN, ERROR, FATAL (**messageType**).
- (opzionalmente) un numero a piacere di messaggi (**messageList**).
- (opzionalmente) la mappa dei parametri in ingresso ad un metodo (**incomingParamMap**).
- (opzionalmente) la mappa dei risultati dell'esecuzione di un metodo (**resultMap**).
- (opzionalmente) la mappa con informazioni di contorno (**infoMap**).
- (opzionalmente, in caso di exception/warning) lista messaggi contenenti le possibili cause (**reasons*List***).
- (opzionalmente, in caso di exception/warning) lista messaggi contenenti le possibili soluzioni suggerite (**fixes*List***).

La lista delle reason e dei fix spesso sono rivolti piu agli utenti congiguratori che agli utenti di geoweb

```
public class GwLogMessage {
    private String messageType;
    private List<String> messageList;
    private Map<String,Object> infoMap; //generic infos map
    private Map<String,Object> incomingParamMap;
    private Map<String,Object> resultMap;
    private List<String> reasonsList;
    private List<String> fixesList;
    ...
}
```

Le istanze della classe GwLogMessage possono essere create tramite costruttore ed utilizzo metodi setters o, piu comodamente tramite la classe **GwLogMessageBuilder** (presente nel package *com.geowebframework.transfer.objects.log*).

Essa permette di concatenare sul costruttore tutta una serie di metodi per poter settarele variabili di una particolare istanza di GwLogMessage. Una volta scelto il set delle caratteristiche, bisognerà invocare sul gwLogMessageBuilder il metodo .build() per ottenere l'istanza di GwLogMessage desiderata.

```
GwLogMessage gwLogMessage = new GwLogMessageBuilder()
    .info()
    .forJava()
    .setLogger(log)
    .addMessage("Hello World")
    .build()
```

Di seguito un esempio che coinvolge le classi sopra esposte Supponiamo che la classe *MyBankService* mi esponga la funzione *withdraw*() che mi permette di prelevare una certa quantita di coins dal totale *ownerCoins*. Supponiamo inoltre che la funzione sia richiamata da una form dove l'utente puo scrivere la quantità di ritirare, e possa vedere l'output in risposta.

```
 class MyBankService {
@Autowired
private GwLogService gwLogService;
private static final Logger log = Logger.getLogger(MyBankService .class);
...
private final String owner;
private int ownerCoins; //number of coins
public boolean withdraw(Integer coins){
    boolean success = false;
    GwLogMessageBuilder gwLogMessageBuilder = null;
    try{
        gwLogMessageBuilder = new GwLogMessageBuilder()
           .info()
           .forJava()
           .setLogger(log)
           .addMessage("MyBankService - withdraw(coins)")
           .addIncomingParamMapEntry("coins", coins);
        if(coins==null) throw new RuntimeException("coins is null!");
        //adding info about owner
        gwLogMessageBuilder.addInfoMapEntry("owner", owner);
        if(ownerCoins<=0){
            String errorMessage = "Impossible to withdraw Exception";
            RuntimeException e = new RuntimeException(errorMessage);
            gwLogMessageBuilder.addReason("Your coins are finished!");
            gwLogMessageBuilder.addFix("Wait until the next salary..");
            throw e;
        }else if(coins>ownerCoins){
            String errorMessage = "Impossible to withdraw Exception";
            RuntimeException e = new RuntimeException(errorMessage);
            gwLogMessageBuilder.addReason(
                "Your are asking a value bigger than "+
                "your own residual coins: "+ownerCoins);
```

```
                gwLogMessageBuilder.addFix(
                    "Try writing in the form a lower value to withdraw"
                );
                throw e;
            }else{
                 ownerCoins -= coins;
                 gwLogMessageBuilder.addInfoMapEntry(
                    "Remaining coint", ownerCoins
                 );
                 if(ownerCoins==0)
                    gwLogMessageBuilder.addMessage("No more coins in bank
now");
                 success = true;
             }
            }catch(Exception e){
                success = false;
                if(gwLogMessageBuilder!=null)
                    gwLogMessageBuilder.error().setThrowable(e);
                //throw e; //throwing e is optional
            }finally{ //always executed
                gwLogMessageBuilder.addResultMapEntry("success", success);
                if(gwLogMessageBuilder!=null)
                    gwLogService.logMessage(gwLogMessageBuilder.build());
            }
            return success;
        }
    }
 }
```

Nell'esempio si vede come il GwLogMessage si sta man mano creando dentro il GwLogMessageBuilder, a seconda delle condizioni il messagggio viene arricchito o meno di messaggi e oggetti accessori. In caso di eccezioni sono aggiunte reasons e fix.

Se ownerCoins=10 e coins=4, l'output è:

```
   dd-MM-yyyy kk:mm:ss,SSS - GEOWEB INFO
   MyBankService - withdraw(coins)
            Incoming Parameters:
                    coins: 4
            Results:
                    success: true
            Related Infos:
                    owner: 'Owner'
                    Remaining coins: 6
```

Se ownerCoins=10 e coins=10, l'output è:

```
  dd-MM-yyyy kk:mm:ss,SSS - GEOWEB INFO
  MyBankService - withdraw(coins)
  No more coins in bank now
      Incoming Parameters:
```

```
                  coins: 10
      Results:
                  success: true
      Related Infos:
                  owner: 'Owner'
                  Remaining coins: 0
```

Se ownerCoins=10 e coins=15, l'output è:

```
   dd-MM-yyyy kk:mm:ss,SSS - GEOWEB ERROR - Impossible to withdraw Exception
   MyBankService - withdraw(coins)
      Incoming Parameters:
                  coins: 15
      Results:
                  success: false
      Related Infos:
                  owner: 'Owner'
      Possible Reasons:
                  Your are asking a value bigger than our own residual
coins: 10
      Suggested quick fix:
                  Try writing in the form a lower value to withdraw
      Stack:
              ....
```

In generale, da notare:

- In generale per tutti i metodi *MapEntry(“key”, value) esiste un equivalente *Map(map) che prende in igresso una direttamente una mappa.
- .forJava() è usato per formattare la stringa risultante dal GwLogMessage al momento del log con i caratteri ritorno a capo e tab specifici per la console java. Questo è il default per GwLogMessage. Esiste un .forHtml() nel caso il messagio sia destinato ad essere visulizzato in una pagina html
- .setThrowable(e) aggiunge in automatico e.getMessage() alla messageList, e logga lo stack
- .setLogger(log) è opzionale, se omesso verra usato un default
- a prescindere da quando aggiungo i vari message, reason, fix ed oggetti al mometo del log dell'istanza di GwLogMessage li vedrò tutti raggruppati per categorie.