ago 31, 17 7:29 **PageAnalyzer.java** Page 1/2

```
1  package ar.fiuba.taller.utils;
2
3  import java.io.IOException;
4  import java.net.URISyntaxException;
5  import java.util.HashMap;
6  import java.util.Iterator;
7  import java.util.Map;
8
9  import javax.swing.text.BadLocationException;
10
11 import org.apache.log4j.Logger;
12 import org.apache.log4j.MDC;
13 import org.jsoup.Jsoup;
14 import org.jsoup.nodes.Document;
15 import org.jsoup.nodes.Element;
16 import org.jsoup.select.Elements;
17
18 import ar.fiuba.taller.loadTestConsole.Constants;
19 import ar.fiuba.taller.loadTestConsole.User;
20
21 public class PageAnalyzer {
22   final static Logger logger = Logger.getLogger(User.class);
23
24   public PageAnalyzer() {
25     MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
26   }
27
28   public Map<String, String> getResources(String response, String url) {
29     Map<String, String> tmpMap = new HashMap<String, String>();
30     Iterator<Element> it = null;
31     Document doc = null;
32     Elements resource = null;
33     String tmpUrl = null;
34     doc = Jsoup.parse(response);
35
36     for (Map.Entry<String, String> entry : Constants.RESOURCE_MAP
37         .entrySet()) {
38       resource = doc.select(entry.getKey());
39       it = resource.iterator();
40       while (it.hasNext()) {
41         tmpUrl = it.next().attr(entry.getValue());
42         if (tmpUrl.indexOf("http") == -1) {
43           tmpUrl = normalizeUrl(url, "last") + "/"
44               + normalizeUrl(tmpUrl, "first");
45         }
46         tmpMap.put(entry.getKey(), tmpUrl);
47       }
48     }
49     return tmpMap;
50   }
51
52   private String normalizeUrl(String url, String place) {
53     String tmpUrl = url.trim();
54
55     if (("").equals(tmpUrl)) {
56       return tmpUrl;
57     }
58     if (place.equals("first")) {
59       while (tmpUrl.substring(0, 1).equals("/")) {
60         tmpUrl = tmpUrl.substring(1, tmpUrl.length());
61       }
62     } else { // last
63       while (tmpUrl.substring(tmpUrl.length() - 1).equals("/")) {
64         tmpUrl = tmpUrl.substring(0, tmpUrl.length() - 1);
65       }
66     }
```

ago 31, 17 7:29 **PageAnalyzer.java** Page 2/2

```
67     return tmpUrl;
68   }
69 }
```

ago 31, 17 7:29 **HttpRequester.java** Page 1/3

```java
package ar.fiuba.taller.utils;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;

public class HttpRequester {

  public HttpRequester() {
  }

  public String doHttpRequest(String method, String url, String headers,
      String data, int timeout) throws IOException {
    String result = null;
    String[] headersArray, tmp;
    Map<String, String> headersMap = new HashMap<String, String>();

    if (headers ≠ null) {
      headersArray = headers.split(";");
      for (int i = 0; i < headersArray.length; i++) {
        tmp = headersArray[i].split(":");
        headersMap.put(tmp[0], tmp[1]);
      }
    }

    if (method.toLowerCase().equals("get")) {
      result = doGet(url, headersMap, data, timeout);
    } else if (method.toLowerCase().equals("post")) {
      result = doPost(url, headersMap, data, timeout);
    } else if (method.toLowerCase().equals("put")) {
      result = doPut(url, headersMap, data, timeout);
    }
    return result;

  }

  // HTTP GET request
  private String doGet(String url, Map<String, String> headers, String data,
      int timeout) throws IOException {
    // Armo la conexion
    URL obj = new URL(url);
    try {
      HttpURLConnection con = (HttpURLConnection) obj.openConnection();
      // optional default is GET
      con.setRequestMethod("GET");
      con.setConnectTimeout(timeout);
      // add request header
      if (¬headers.isEmpty()) {
        for (Map.Entry<String, String> entry : headers.entrySet()) {
          con.setRequestProperty(entry.getKey(), entry.getValue());
        }
      }

      BufferedReader in = new BufferedReader(
          new InputStreamReader(con.getInputStream()));
      String inputLine;
      StringBuffer response = new StringBuffer();
      while ((inputLine = in.readLine()) ≠ null) {
        response.append(inputLine);
      }
      in.close();
```

ago 31, 17 7:29 **HttpRequester.java** Page 2/3

```java
      if ("".equals(response.toString())) {
        return null;
      } else {
        return response.toString();
      }
    } catch (java.net.SocketTimeoutException e) {
      return null;
    }
    // } catch (java.io.IOException e) {
    // return null;
    // }
    // print result
  }

  // HTTP POST request
  private String doPost(String url, Map<String, String> headers, String data,
      int timeout) throws IOException {
    URL obj = new URL(url);
    try {
      HttpURLConnection con = (HttpURLConnection) obj.openConnection();

      // add reuqest method
      con.setRequestMethod("POST");
      con.setConnectTimeout(timeout);
      // add reuqest header
      if (¬headers.isEmpty()) {
        for (Map.Entry<String, String> entry : headers.entrySet()) {
          con.setRequestProperty(entry.getKey(), entry.getValue());
        }
      }

      // Send post request
      con.setDoOutput(true);
      DataOutputStream wr = new DataOutputStream(con.getOutputStream());
      wr.writeBytes(data);
      wr.flush();
      wr.close();

      BufferedReader in = new BufferedReader(
          new InputStreamReader(con.getInputStream()));
      String inputLine;
      StringBuffer response = new StringBuffer();

      while ((inputLine = in.readLine()) ≠ null) {
        response.append(inputLine);
      }
      in.close();

      // print result
      return response.toString();
    } catch (java.net.SocketTimeoutException e) {
      return null;
    }
  }

  // HTTP PUT request
  private String doPut(String url, Map<String, String> headers, String data,
      int timeout) throws IOException {
    URL obj = new URL(url);
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();
    try {
      // add reuqest method
      con.setRequestMethod("PUT");
      con.setConnectTimeout(timeout);

      // add reuqest header
```

ago 31, 17 7:29 **HttpRequester.java** Page 3/3

```
133         if (¬headers.isEmpty()) {
134           for (Map.Entry<String, String> entry : headers.entrySet()) {
135             con.setRequestProperty(entry.getKey(), entry.getValue());
136           }
137         }
138
139         // Send post request
140         con.setDoOutput(true);
141         DataOutputStream wr = new DataOutputStream(con.getOutputStream());
142         wr.writeBytes(data);
143         wr.flush();
144         wr.close();
145
146         BufferedReader in = new BufferedReader(
147             new InputStreamReader(con.getInputStream()));
148         String inputLine;
149         StringBuffer response = new StringBuffer();
150
151         while ((inputLine = in.readLine()) ≠ null) {
152           response.append(inputLine);
153         }
154         in.close();
155
156         // print result
157         return response.toString();
158       } catch (java.net.SocketTimeoutException e) {
159         return null;
160       }
161     }
162
163 }
```

ago 27, 17 16:52 **FileWatcher.java** Page 1/1

```
1  package ar.fiuba.taller.utils;
2
3  import java.util.*;
4  import java.io.*;
5
6  public abstract class FileWatcher extends TimerTask {
7    private long timeStamp;
8    private File file;
9
10   public FileWatcher( File file ) {
11     this.file = file;
12     this.timeStamp = file.lastModified();
13   }
14
15   public final void run() {
16     long timeStamp = file.lastModified();
17
18     if( this.timeStamp ≠ timeStamp ) {
19       this.timeStamp = timeStamp;
20       onChange(file);
21     }
22   }
23
24   protected abstract void onChange( File file );
25 }
```

ago 21, 17 20:27 **UserStat.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

public class UserStat extends SummaryStat {
  private int usersAmount;

  @Override
  public void updateSumary(Summary summary) {
    summary.setUsers(usersAmount);

  }

  public UserStat(int usersAmount) {
    super();
    this.usersAmount = usersAmount;
  }

}
```

ago 31, 17 7:27 **UsersController.java** Page 1/3

```java
package ar.fiuba.taller.loadTestConsole;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.atomic.AtomicInteger;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;

import ar.fiuba.taller.loadTestConsole.Constants.REPORT_EVENT;

public class UsersController implements Runnable {

  final static Logger logger = Logger.getLogger(UsersController.class);
  private int maxUsers;
  private Map<Integer, Integer> usersPatternMap;
  private Map<String, String> propertiesMap;
  private AtomicInteger patternTime;
  private ArrayBlockingQueue<SummaryStat> summaryQueue;
  private ArrayBlockingQueue<REPORT_EVENT> reportQueue;
  private List<Future<User>> futures = null;

  public UsersController(Map<String, String> propertiesMap,
      Map<Integer, Integer> usersPatternMap, AtomicInteger patternTime,
      ArrayBlockingQueue<SummaryStat> summaryQueue,
      ArrayBlockingQueue<REPORT_EVENT> reportQueue) {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
    this.propertiesMap = propertiesMap;
    this.maxUsers = Integer
        .parseInt(this.propertiesMap.get(Constants.MAX_USERS));
    this.usersPatternMap = usersPatternMap;
    this.patternTime = patternTime;
    this.summaryQueue = summaryQueue;
    this.reportQueue = reportQueue;
    this.futures = new ArrayList<>();
  }

  @Override
  public void run() {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
    int totalUsersCount = 0; // Usuarios totales corriendo
    int sleepTime = 0; // Tiempo a esperar
    int deltaUsers = 0; // Usuarios que se deben agregar o quitar
    int oldTime = 0; // Tiempo del pulso anterior
    ExecutorService executorService = Executors
        .newFixedThreadPool(maxUsers);

    Iterator<Map.Entry<Integer, Integer>> it = usersPatternMap.entrySet()
        .iterator();
    Map.Entry<Integer, Integer> pair;

    logger.info("Iniciando UsersController");

    if (it.hasNext()) {
      pair = it.next();
      deltaUsers = pair.getValue() - totalUsersCount;
      oldTime = pair.getKey();
    }

    try {
```

```
67        while (¬Thread.interrupted()) {
68          logger.info(
69              "Usuarios corriendo en el pool: " + totalUsersCount);
70          logger.info("Usuarios que se deben ingresar al pool: "
71              + deltaUsers);
72          totalUsersCount += updateUsers(totalUsersCount, deltaUsers,
73              executorService);
74          logger.info("Usuarios ingresados");
75          if (it.hasNext()) {
76            pair = it.next();
77            if (pair.getKey() > oldTime) {
78              sleepTime = pair.getKey() − oldTime;
79            }
80            deltaUsers = pair.getValue() − totalUsersCount;
81            oldTime = pair.getKey();
82            patternTime.set(oldTime);
83            logger.debug("Valores para la proxima corrida: "
84                + pair.getKey() + "−" + pair.getValue());
85          } else {
86            deltaUsers = 0;
87          }
88          logger.info("Usuarios totales: " + totalUsersCount);
89          summaryQueue.put(new UserStat(totalUsersCount));
90          logger.info(
91              "Tiempo a dormir hasta el proximo pulso: " + sleepTime);
92          Thread.sleep(sleepTime * Constants.SLEEP_UNIT);
93        }
94      } catch (InterruptedException e) {
95        logger.info(
96            "Senial de interrupcion recibida. Eliminado los Usuarios.");
97        executorService.shutdownNow();
98      }
99    }
100
101   /*
102    * Actualiza el pool de threads con los usuarios pasados Retorna la cantidad
103    * de usuarios agregados o eliminados
104    */
105   @SuppressWarnings("unchecked")
106   private int updateUsers(int totalUsersCount, int deltaUsers,
107       ExecutorService executorService) {
108     int usersToAdd = 0;
109
110     if (totalUsersCount + deltaUsers ≥ maxUsers) {
111       usersToAdd = maxUsers − totalUsersCount;
112     } else if (totalUsersCount + deltaUsers < 0) {
113       usersToAdd = 0;
114     } else {
115       usersToAdd = deltaUsers;
116     }
117
118     logger.info("Usuarios a agregar: " + usersToAdd);
119     if (usersToAdd > 0) {
120       logger.info("Agregando usuarios");
121       // Disparo los users
122       for (int i = 0; i < usersToAdd; i++) {
123         futures.add((Future<User>) executorService.submit(
124             new User(propertiesMap, summaryQueue, reportQueue)));
125       }
126
127     } else if (usersToAdd < 0) {
128       logger.info("Eliminando usuarios");
129       int tmpUsersToAdd = Math.abs(usersToAdd);
130       Iterator<Future<User>> it = futures.iterator();
131       Future<User> f;
132       for (int i = 0; i < tmpUsersToAdd; i++) {
```



```
133         f = it.next();
134         f.cancel(true);
135         it.remove();
136         logger.debug("Usuario cancelado");
137       }
138     }
139     return usersToAdd;
140   }
141 }
```

ago 31, 17 7:29 **User.java** Page 1/3

```
1  package ar.fiuba.taller.loadTestConsole;
2
3  import java.io.FileReader;
4  import java.io.IOException;
5  import java.util.HashSet;
6  import java.util.Iterator;
7  import java.util.List;
8  import java.util.Map;
9  import java.util.Set;
10 import java.util.concurrent.ArrayBlockingQueue;
11 import java.util.concurrent.Callable;
12 import java.util.concurrent.ExecutionException;
13 import java.util.concurrent.ExecutorService;
14 import java.util.concurrent.Executors;
15 import java.util.concurrent.Future;
16
17 import org.apache.log4j.Logger;
18 import org.apache.log4j.MDC;
19 import org.json.simple.JSONArray;
20 import org.json.simple.JSONObject;
21 import org.json.simple.parser.JSONParser;
22 import org.json.simple.parser.ParseException;
23
24 import ar.fiuba.taller.loadTestConsole.Constants.REPORT_EVENT;
25 import ar.fiuba.taller.utils.HttpRequester;
26 import ar.fiuba.taller.utils.PageAnalyzer;
27
28 public class User implements Runnable {
29   final static Logger logger = Logger.getLogger(User.class);
30   private Map<String, String> propertiesMap;
31   private ArrayBlockingQueue<SummaryStat> summaryQueue;
32   private ArrayBlockingQueue<REPORT_EVENT> reportQueue;
33
34   public User(Map<String, String> propertiesMap,
35       ArrayBlockingQueue<SummaryStat> summaryQueue,
36       ArrayBlockingQueue<REPORT_EVENT> reportQueue) {
37     MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
38     this.summaryQueue = summaryQueue;
39     this.reportQueue = reportQueue;
40     this.propertiesMap = propertiesMap;
41   }
42
43   @SuppressWarnings("unchecked")
44   @Override
45   public void run() {
46     long time_end, time_start, avgTime = 0, successResponse = 0,
47         failedResponse = 0;
48     String response = null;
49     Map<String, String> resourceMap = null;
50     Set<Callable<Downloader>> downloadersSet = new HashSet<Callable<Downloader>>
   ();
51     ExecutorService executorService = Executors.newFixedThreadPool(
52         Integer.parseInt(propertiesMap.get(Constants.MAX_DOWNLOADERS)));
53     JSONParser parser = new JSONParser();
54     HttpRequester httpRequester = new HttpRequester();
55     Object objScript = null;
56     JSONObject objStep = null;
57     JSONArray stepsArray = null;
58     List<Future<Downloader>> futures = null;
59     PageAnalyzer pageAnalyzer = new PageAnalyzer();
60     Iterator<JSONObject> it = null;
61
62     logger.info("Iniciando usuario");
63     try {
64       reportQueue.put(REPORT_EVENT.SCRIPT_EXECUTING);
65       objScript = parser.parse(
```

ago 31, 17 7:29 **User.java** Page 2/3

```
66           new FileReader(propertiesMap.get(Constants.SCRIPT_FILE)));
67       stepsArray = (JSONArray) ((JSONObject) objScript).get("steps");
68       it = stepsArray.iterator();
69
70       while (¬Thread.interrupted()) {
71         if (it ≡ null ∨ ¬it.hasNext()) {
72           avgTime = 0;
73           successResponse = 0;
74           failedResponse = 0;
75           downloadersSet.clear();
76           it = stepsArray.iterator();
77         }
78         objStep = it.next();
79         logger.info("Siguiente url a analizar: "
80             + (String) objStep.get("url"));
81         logger.info("Metodo: " + (String) objStep.get("method"));
82         logger.info("headers: " + (String) objStep.get("headers"));
83         logger.info("Body: " + (String) objStep.get("body"));
84         time_start = System.currentTimeMillis();
85         response = httpRequester.doHttpRequest(
86             (String) objStep.get("method"),
87             (String) objStep.get("url"),
88             (String) objStep.get("headers"),
89             (String) objStep.get("body"),
90             Integer.parseInt(
91                 propertiesMap.get(Constants.HTTP_TIMEOUT))
92                 * Constants.SLEEP_UNIT);
93         logger.debug("Request listo");
94         time_end = System.currentTimeMillis();
95         avgTime = time_end − time_start;
96         if (response ≡ null) {
97           failedResponse++;
98         } else {
99           successResponse++;
100          resourceMap = pageAnalyzer.getResources(response,
101              (String) objStep.get("url"));
102          reportQueue.put(REPORT_EVENT.URL_ANALYZED);
103          for (Map.Entry<String, String> entry : resourceMap
104              .entrySet()) {
105            logger.debug("tipo: " + entry.getKey());
106            logger.debug("recurso: " + entry.getValue());
107            downloadersSet.add(new Downloader(reportQueue,
108                entry.getValue(), entry.getKey(), propertiesMap,
109                summaryQueue));
110          }
111          futures = executorService.invokeAll(downloadersSet);
112          logger.info("Esperando Downloaders");
113          for (Future<Downloader> future : futures) {
114            future.get();
115          }
116        }
117        summaryQueue.put(new RequestStat(successResponse,
118            failedResponse, avgTime));
119      }
120    } catch (IOException | InterruptedException | ExecutionException e) {
121      // Do nothing
122    } catch (ParseException e) {
123      try {
124        throw new ParseException(0);
125      } catch (ParseException e1) {
126        // Do nothing
127      }
128    } finally {
129      try {
130        logger.info("User cancelado. Eliminando downloaders.");
131        executorService.shutdownNow();
```

ago 31, 17 7:29 **User.java** Page 3/3

```
132           reportQueue.put(REPORT_EVENT.SCRIPT_EXECUTED);
133         } catch (InterruptedException e1) {
134           // Do nothing
135         }
136       }
137     }
138   }
```

ago 21, 17 20:24 **SummaryStat.java** Page 1/1

```
1  package ar.fiuba.taller.loadTestConsole;
2
3  public abstract class SummaryStat {
4
5    public abstract void updateSumary(Summary summary);
6  }
```

ago 31, 17 7:29 **SummaryPrinter.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

import java.util.Map;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;

public class SummaryPrinter implements Runnable {
  private Summary summary;
  private Map<String, String> propertiesMap;
  final static Logger logger = Logger.getLogger(SummaryPrinter.class);

  public SummaryPrinter(Summary summary, Map<String, String> propertiesMap) {
    this.summary = summary;
    this.propertiesMap = propertiesMap;
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
  }

  public void run() {
    logger.info("Iniciando SummaryPrinter");
    final String ANSI_CLS = "\u001b[2J";
    final String ANSI_HOME = "\u001b[H";
    while (¬Thread.interrupted()) {
      // Limpio la pantalla
      System.out.print(ANSI_CLS + ANSI_HOME);
      System.out.flush();

      // Imprimo el resumen
      System.out.printf(
          "Load Test Console: Resumen de ejecucion%n−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−%nTiempo de descarga promedio...: %d ms%nRequests exitosos.............: %d / %d %nRequests fallidos.............: %d / %d %nCantidad de usuarios..........: %d%nPresione ^C para terminar...%n",
          summary.getAvgDownloadTime(),
          summary.getSuccessfullrequest(), summary.getTotalRequests(),
          summary.getFailedrequest(), summary.getTotalRequests(),
          summary.getUsers());
      try {
        Thread.sleep(Long
            .parseLong(propertiesMap.get(Constants.SUMMARY_TIMEOUT))
            * Constants.SLEEP_UNIT);
      } catch (InterruptedException e) {
        logger.info("SummaryPrinter interrumpido");
      }
    }
    logger.info("SummaryPrinter finalizado");
  }
}
```

ago 27, 17 10:34 **Summary.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

public class Summary {
  private long avgDownloadTime;
  private long successfullrequest;
  private long failedrequest;
  private Integer users;

  public Summary() {
    avgDownloadTime = 0;
    successfullrequest = 0;
    failedrequest = 0;
    users = 0;
  }

  public synchronized long getSuccessfullrequest() {
    return successfullrequest;
  }

  public synchronized void incSuccessfullrequest(long count) {
    successfullrequest += count;
  }

  public synchronized long getFailedrequest() {
    return failedrequest;
  }

  public synchronized void incFailedrequest(long count) {
    failedrequest += count;
  }

  public synchronized Integer getUsers() {
    return users;
  }

  public synchronized void setUsers(Integer users) {
    this.users = users;
  }

  public synchronized long getAvgDownloadTime() {
    return avgDownloadTime;
  }

  public synchronized void updateAvgDownloadTime(long time) {
    avgDownloadTime = (avgDownloadTime + time)/2;
  }

  public synchronized long getTotalRequests() {
    return successfullrequest + failedrequest;
  }
}
```

ago 27, 17 8:56 **SummaryController.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

import java.util.concurrent.ArrayBlockingQueue;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;

public class SummaryController implements Runnable {
  final static Logger logger = Logger.getLogger(SummaryController.class);
  private ArrayBlockingQueue<SummaryStat> summaryQueue;
  private Summary summary;


  public SummaryController(ArrayBlockingQueue<SummaryStat> summaryQueue,
      Summary summary) {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
    this.summaryQueue = summaryQueue;
    this.summary = summary;
  }

  public void run() {
    logger.info("Se inicia el summary controller");
    SummaryStat summaryStat = null;
    while(¬Thread.interrupted()) {
      try {
        summaryStat = summaryQueue.take();
      } catch (InterruptedException e) {
        logger.info("summary controller interrumpido.");
      }
      summaryStat.updateSumary(summary);
    }
    logger.info("summary controller finalizado.");
  }
}
```

ago 27, 17 10:34 **RequestStat.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

public class RequestStat extends SummaryStat {
  private long successfullRequest;
  private long failedRequest;
  private long avgDownloadTime;

  public RequestStat(long successfullRequest, long failedRequest,
      long avgDownloadTime) {
    super();
    this.successfullRequest = successfullRequest;
    this.failedRequest = failedRequest;
    this.avgDownloadTime = avgDownloadTime;
  }

  @Override
  public void updateSumary(Summary summary) {
    summary.incSuccessfullrequest(successfullRequest);
    summary.incFailedrequest(failedRequest);
    summary.updateAvgDownloadTime(avgDownloadTime);
  }

}
```

ago 31, 17 7:28 **ReportPrinter.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Map;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;

public class ReportPrinter implements Runnable {

  private Report report;
  private Map<String, String> propertiesMap;
  final static Logger logger = Logger.getLogger(ReportPrinter.class);

  public ReportPrinter(Report report, Map<String, String> propertiesMap) {
    this.report = report;
    this.propertiesMap = propertiesMap;
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
  }

  public void run() {
    logger.info("Iniciando Monitor");
    while (¬Thread.interrupted()) {
      try {
        PrintWriter writer = new PrintWriter(
            propertiesMap.get(Constants.REPORT_FILE), "UTF-8");
        writer.printf(
            "Load Test Console: Monitor de reportes%n----------------------------------------%n
URLs analizadas.....: %d%nSCRIPTS descargados.....: %d%nLINKS descargados.....: %d%nI
MGs descargadas.....: %d%nHilos ejecutando script.....: %d%nHilos descargando recurso.....: %d%n",
            report.getAnalyzedUrl(), report.getDownloadedScripts(),
            report.getDownloadedLinks(),
            report.getDownloadedImages(),
            report.getExecutionScriptThreads(),
            report.getDownloadResourceThreads());
        writer.close();
        Thread.sleep(Integer
            .parseInt(propertiesMap.get(Constants.REPORT_TIMEOUT))
            * Constants.SLEEP_UNIT);
      } catch (IOException e) {
        logger.error("No se pudo abrir el report file");
      } catch (NumberFormatException e) {
        logger.error("Tiempo del sleep mal seteado");
      } catch (InterruptedException e) {
        logger.info("Report interrumpido");
      }
    }
    logger.info("Finalizando Monitor");
  }
}
```

ago 27, 17 9:48 **Report.java** Page 1/2

```java
package ar.fiuba.taller.loadTestConsole;

public class Report {

  private Integer analyzedUrl;
  private Integer downloadedScripts;
  private Integer downloadedLinks;
  private Integer downloadedImages;
  private Integer executionScriptThreads;
  private Integer downloadResourceThreads;

  public synchronized Integer getAnalyzedUrl() {
    return analyzedUrl;
  }

  public synchronized void incAnalyzedUrl() {
    analyzedUrl++;
  }

  public synchronized Integer getDownloadedScripts() {
    return downloadedScripts;
  }

  public synchronized void incDownloadedScripts() {
    downloadedScripts++;
  }

  public synchronized Integer getDownloadedLinks() {
    return downloadedLinks;
  }

  public synchronized void incDownloadedLinks() {
    downloadedLinks++;
  }

  public synchronized Integer getDownloadedImages() {
    return downloadedImages;
  }

  public synchronized void incDownloadedImages() {
    downloadedImages++;
  }

  public synchronized Integer getExecutionScriptThreads() {
    return executionScriptThreads;
  }

  public synchronized void incExecutionScriptThreads() {
    executionScriptThreads++;
  }

  public synchronized void decExecutionScriptThreads() {
    executionScriptThreads--;
  }

  public synchronized Integer getDownloadResourceThreads() {
    return downloadResourceThreads;
  }

  public synchronized void incDownloadResourceThreads() {
    downloadResourceThreads++;
  }

  public synchronized void decDownloadResourceThreads() {
    downloadResourceThreads--;
  }
```

ago 27, 17 9:48 **Report.java** Page 2/2

```
67
68    public Report() {
69      analyzedUrl = 0;
70      downloadedScripts = 0;
71      downloadedLinks = 0;
72      downloadedImages = 0;
73      executionScriptThreads = 0;
74      downloadResourceThreads = 0;
75    }
76
77  }
```

ago 27, 17 12:57 **ReportController.java** Page 1/1

```
1   package ar.fiuba.taller.loadTestConsole;
2
3   import java.util.concurrent.ArrayBlockingQueue;
4
5   import org.apache.log4j.Logger;
6   import org.apache.log4j.MDC;
7
8   import ar.fiuba.taller.loadTestConsole.Constants.REPORT_EVENT;
9
10  public class ReportController implements Runnable {
11
12    private ArrayBlockingQueue<REPORT_EVENT> reportQueue;
13    private Report report;
14    final static Logger logger = Logger.getLogger(ReportController.class);
15
16    public ReportController(ArrayBlockingQueue<REPORT_EVENT> reportQueue,
17        Report report) {
18      MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
19      this.reportQueue = reportQueue;
20      this.report = report;
21    }
22
23    public void run() {
24      logger.info("Se inicia el report controller");
25      REPORT_EVENT reportEvent;
26
27      while(¬Thread.interrupted()) {
28        try {
29          reportEvent = reportQueue.take();
30          switch (reportEvent) {
31          case URL_ANALYZED:
32            report.incAnalyzedUrl();
33            break;
34          case SCRIPT_DOWNLOADED:
35            report.incDownloadedScripts();
36            break;
37          case LINK_DOWNLOADED:
38            report.incDownloadedLinks();
39            break;
40          case IMG_DOWNLOADED:
41            report.incDownloadedImages();
42            break;
43          case SCRIPT_EXECUTING:
44            report.incExecutionScriptThreads();
45            break;
46          case SCRIPT_EXECUTED:
47            report.decExecutionScriptThreads();
48            break;
49          case RESOURCE_DOWNLOAD:
50            report.incDownloadResourceThreads();
51            break;
52          case RESOURCE_DOWNLOADED:
53            report.decDownloadResourceThreads();
54            break;
55          default:
56            break;
57          }
58        } catch (InterruptedException e) {
59          logger.info("Report Controller interrumpido.");
60        }
61      }
62      logger.info("Report Controller finalizado.");
63    }
64  }
```

ago 21, 17 10:39 **PatternFileWatcher.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

import java.io.File;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.Semaphore;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;

import ar.fiuba.taller.utils.FileWatcher;


public class PatternFileWatcher implements Runnable {
  private Semaphore fileChangeSem;
  private String patternFilePath;
  private int pullTime;
  final static Logger logger = Logger.getLogger(PatternFileWatcher.class);

  public PatternFileWatcher(String patternFilePath, Semaphore fileChangeSem,
      int pullTime) {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
    this.patternFilePath = patternFilePath;
    this.fileChangeSem = fileChangeSem;
    this.pullTime = pullTime;
  }

  @Override
  public void run() {
    // monitor a single file
    logger.info("Iniciando PatternFileWatcher");
    logger.debug( "patternFilePath -> " + patternFilePath);
    logger.debug( "pullTime -> " + pullTime);
    TimerTask task = new FileWatcher( new File(patternFilePath) ) {
      protected void onChange( File file ) {
        // here we code the action on a change
        logger.info("Ha cambiado el archivo del patron de usuarios");
        fileChangeSem.release();
      }
    };

    Timer timer = new Timer();
    // repeat the check every second
    timer.schedule( task , new Date(), pullTime );
  }

}
```

ago 21, 17 23:03 **Main.java** Page 1/1

```java
package ar.fiuba.taller.loadTestConsole;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;

public class Main {
  final static Logger logger = Logger.getLogger(Main.class);

  public static void main(String[] args) {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));

    LoadTestConsole loadTestConsole;
    try {
      loadTestConsole = new LoadTestConsole();
      logger.info("[*] Se inicia una nueva instancia de LoadTestConsole");
      loadTestConsole.start();
      logger.info("[*] Finaliza la instancia de LoadTestConsole");
    } catch (Exception e) {
      System.exit(Constants.EXIT_FAILURE);
    }
    System.exit(Constants.EXIT_SUCCESS);
  }
}
```

ago 31, 17 7:28 **LoadTestConsole.java** Page 1/3

```java
package ar.fiuba.taller.loadTestConsole;

import ar.fiuba.taller.loadTestConsole.Constants.REPORT_EVENT;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;
import java.util.InputMismatchException;
import java.util.Map;
import java.util.Properties;
import java.util.Scanner;
import java.util.TreeMap;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.Semaphore;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.regex.Pattern;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;

public class LoadTestConsole {
  private Map<String, String> propertiesMap;
  private Map<Integer, Integer> usersPatternMap;

  final static Logger logger = Logger.getLogger(LoadTestConsole.class);

  public LoadTestConsole() throws IOException {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
    // Cargo la configuracion del archivo de properties
    loadProperties();
  }

  public void start() throws FileNotFoundException {
    Semaphore fileChangeSem = new Semaphore(0);
    ArrayBlockingQueue<SummaryStat> summaryQueue =
        new ArrayBlockingQueue<SummaryStat>(
        Integer.parseInt(
            propertiesMap.get(Constants.SUMMARY_QUEUE_SIZE)));
    ArrayBlockingQueue<REPORT_EVENT> reportQueue =
        new ArrayBlockingQueue<REPORT_EVENT>(
        Integer.parseInt(
            propertiesMap.get(Constants.REPORT_QUEUE_SIZE)));
    Summary summary = new Summary();
    Report report = new Report();
    AtomicInteger patternTime = new AtomicInteger(0);

    // Creo los threads
    Thread usersControllerThread = null;
    Thread patternFileWatcherThread = new Thread(new PatternFileWatcher(
        propertiesMap.get(Constants.USERS_PATTERN_FILE), fileChangeSem,
        Integer.parseInt(
            propertiesMap.get(Constants.FILE_WATCHER_TIMEOUT))));
    Thread summaryControllerThread = new Thread(
        new SummaryController(summaryQueue, summary));
    Thread summaryPrinterThread = new Thread(
        new SummaryPrinter(summary, propertiesMap));
    Thread reportControllerThread = new Thread(
        new ReportController(reportQueue, report));
    Thread reportPrinterThread = new Thread(
        new ReportPrinter(report, propertiesMap));

    logger.info("Iniciando LoadTestConsole");
    logger.info("Iniciando los threads");
    patternFileWatcherThread.start();
    summaryControllerThread.start();
    summaryPrinterThread.start();
```

ago 31, 17 7:28 **LoadTestConsole.java** Page 2/3

```java
    reportControllerThread.start();
    reportPrinterThread.start();

    while (¬Thread.interrupted()) {
      logger.info("Cargando patron de usuarios");
      loadUserPattern(patternTime);
      logger.info("Disparando el usersControllerThread");
      usersControllerThread = new Thread(
          new UsersController(propertiesMap, usersPatternMap,
              patternTime, summaryQueue, reportQueue));
      usersControllerThread.start();
      try {
        // Me quedo esperando hasta que cambie el archivo
        logger.info("Esperando hasta que cambie el archivo");
        fileChangeSem.acquire();
      } catch (InterruptedException e) {
        // Do nothing
        logger.error("No se ha podido tomar el semaforo");
      }
      logger.info(
          "Cambio el archivo. Interrumpiendo el usersControllerThread");
      System.out.println(
          "Cambio el archvio. Recargando patron de usuarios...");
      usersControllerThread.interrupt();
      try {
        usersControllerThread.join();
      } catch (InterruptedException e) {
        // Do nothing
      }
    }
  }

  private void loadProperties() throws IOException {
    logger.info("Cargando configuracion");
    propertiesMap = new HashMap<String, String>();
    Properties properties = new Properties();
    try {
      properties.load(Thread.currentThread().getContextClassLoader()
          .getResourceAsStream(Constants.PROPERTIES_FILE));
    } catch (IOException e) {
      System.err.println(
          "No ha sido posible cargar el archivo de propiedades");
      throw new IOException();
    }
    for (String key : properties.stringPropertyNames()) {
      String value = properties.getProperty(key);
      propertiesMap.put(key, value);
      logger.debug("Parametro cargado: " + key + "->" + value);
    }
  }

  private void loadUserPattern(AtomicInteger patternTime)
      throws FileNotFoundException {
    File file = new File(propertiesMap.get(Constants.USERS_PATTERN_FILE));
    int nextIntTime, nextIntUser;
    if (file ≡ null ∨ ¬file.canRead()) {
      System.out.println(
          "No se ha podido abrir el archivo con el patron de usuarios.");
    }
    usersPatternMap = new TreeMap<Integer, Integer>();
    @SuppressWarnings("resource")
    final Scanner s = new Scanner(file)
        .useDelimiter(Pattern.compile("(\\n)|:"));
    try {
      while (s.hasNext()) {
        nextIntTime = s.nextInt();
```

ago 31, 17 7:28 **LoadTestConsole.java** Page 3/3

```
133           nextIntUser = s.nextInt();
134           logger.debug(
135               "Tiempo desde el cual cargar: " + patternTime.get());
136           if (nextIntTime ≥ patternTime.get()) {
137             usersPatternMap.put(nextIntTime, nextIntUser);
138             logger.debug("Patron cargado: " + nextIntTime + ","
139                 + nextIntUser);
140           }
141         }
142       } catch (InputMismatchException e) {
143         // Do nothing
144       }
145     }
146   }
```

ago 31, 17 7:27 **Downloader.java** Page 1/2

```
1   package ar.fiuba.taller.loadTestConsole;
2
3   import java.util.Map;
4   import java.util.concurrent.ArrayBlockingQueue;
5   import java.util.concurrent.Callable;
6   import org.apache.log4j.Logger;
7   import org.apache.log4j.MDC;
8
9   import ar.fiuba.taller.loadTestConsole.Constants.REPORT_EVENT;
10  import ar.fiuba.taller.utils.HttpRequester;
11
12  public class Downloader implements Callable {
13
14    final static Logger logger = Logger.getLogger(Downloader.class);
15    private ArrayBlockingQueue<REPORT_EVENT> reportQueue;
16    private ArrayBlockingQueue<SummaryStat> summaryQueue;
17    private String url;
18    private String type;
19    private Map<String, String> propertiesMap;
20
21    public Downloader(ArrayBlockingQueue<REPORT_EVENT> reportQueue, String url,
22        String type, Map<String, String> propertiesMap,
23        ArrayBlockingQueue<SummaryStat> summaryQueue) {
24      MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
25      this.reportQueue = reportQueue;
26      this.summaryQueue = summaryQueue;
27      this.url = url;
28      this.type = type;
29      this.propertiesMap = propertiesMap;
30    }
31
32    @Override
33    public Object call() {
34      long time_start, time_end, time_elapsed = 0;
35      String response = null;
36      HttpRequester httpRequester = new HttpRequester();
37      int successResponse = 0, failedResponse = 0;
38
39      logger.info("Iniciando Downloader.");
40      logger.info("Url a descargar: " + url);
41      logger.info("Tipo de recurso: " + type);
42      try {
43        reportQueue.put(REPORT_EVENT.RESOURCE_DOWNLOAD);
44        time_start = System.currentTimeMillis();
45        response = httpRequester.doHttpRequest("get", url, null, null,
46            Integer.parseInt(
47                propertiesMap.get(Constants.HTTP_TIMEOUT)));
48        time_end = System.currentTimeMillis();
49        time_elapsed = time_end − time_start;
50        reportQueue.put(Constants.TYPE_RESOURCE_MAP.get(type));
51      } catch (Exception e) {
52        // Do nothing
53      } finally {
54        try {
55          if (response ≡ null) {
56            successResponse++;
57          } else {
58            failedResponse++;
59          }
60          summaryQueue.put(new RequestStat(successResponse,
61              failedResponse, time_elapsed));
62          reportQueue.put(REPORT_EVENT.RESOURCE_DOWNLOADED);
63        } catch (InterruptedException e) {
64          // Do nothing
65        }
66      }
```

ago 31, 17 7:27 **Downloader.java** Page 2/2

```
67        logger.info("Downloader terminado.");
68        return null;
69      }
70    }
```

ago 31, 17 7:27 **Constants.java** Page 1/1

```
1   package ar.fiuba.taller.loadTestConsole;
2
3   import java.util.HashMap;
4   import java.util.Map;
5
6   public final class Constants {
7     public static final String PROPERTIES_FILE = "configuration.properties";
8     public static final String MAX_USERS = "max.users";
9     public static final String MAX_DOWNLOADERS = "max.downloaders";
10    public static final String SCRIPT_FILE = "script.file";
11    public static final String REPORT_FILE = "report.file";
12    public static final String SUMMARY_TIMEOUT = "summary.timeout";
13    public static final String REPORT_TIMEOUT = "report.timeout";
14    public static final String HTTP_TIMEOUT = "http.timeout";
15    public static final String FILE_WATCHER_TIMEOUT = "file.watcher.timeout";
16    public static final String USERS_PATTERN_FILE = "users.pattern.file";
17    public static final String SUMMARY_QUEUE_SIZE = "summary.queue.size";
18    public static final String REPORT_QUEUE_SIZE = "report.queue.size";
19
20    public static final int EXIT_SUCCESS = 0;
21    public static final int EXIT_FAILURE = 1;
22    public static final int SLEEP_UNIT = 1000;
23
24    public static final Map<String, String> RESOURCE_MAP;
25    static {
26      RESOURCE_MAP = new HashMap<String, String>();
27      RESOURCE_MAP.put("LINK", "href");
28      RESOURCE_MAP.put("SCRIPT", "src");
29      RESOURCE_MAP.put("IMG", "src");
30    }
31
32    public static enum REPORT_EVENT {
33      URL_ANALYZED, SCRIPT_DOWNLOADED, LINK_DOWNLOADED, IMG_DOWNLOADED,
34      SCRIPT_EXECUTING, SCRIPT_EXECUTED, RESOURCE_DOWNLOAD,
35      RESOURCE_DOWNLOADED
36    };
37
38    public static final Map<String, REPORT_EVENT> TYPE_RESOURCE_MAP;
39    static {
40      TYPE_RESOURCE_MAP = new HashMap<String, REPORT_EVENT>();
41      TYPE_RESOURCE_MAP.put("LINK", REPORT_EVENT.LINK_DOWNLOADED);
42      TYPE_RESOURCE_MAP.put("SCRIPT", REPORT_EVENT.SCRIPT_DOWNLOADED);
43      TYPE_RESOURCE_MAP.put("IMG", REPORT_EVENT.IMG_DOWNLOADED);
44    }
45  }
```

ago 31, 17 7:30 **Table of Content** Page 1/1

Table of Contents