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

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

import java.io.IOException;
import java.net.URISyntaxException;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

import javax.swing.text.BadLocationException;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

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

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

  public PageAnalyzer() {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
  }

  public Map<String, String> getResources(String response, String url) {
    Map<String, String> tmpMap = new HashMap<String, String>();
    Iterator<Element> it = null;
    Document doc = null;
    Elements resource = null;
    String tmpUrl = null;
    doc = Jsoup.parse(response);

    for (Map.Entry<String, String> entry : Constants.RESOURCE_MAP
        .entrySet()) {
      resource = doc.select(entry.getKey());
      it = resource.iterator();
      while (it.hasNext()) {
        tmpUrl = it.next().attr(entry.getValue());
        if (tmpUrl.indexOf("http") == -1) {
          tmpUrl = normalizeUrl(url, "last") + "/"
              + normalizeUrl(tmpUrl, "first");
        }
        tmpMap.put(entry.getKey(), tmpUrl);
      }
    }
    return tmpMap;
  }

  private String normalizeUrl(String url, String place) {
    String tmpUrl = url.trim();

    if (("").equals(tmpUrl)) {
      return tmpUrl;
    }
    if (place.equals("first")) {
      while (tmpUrl.substring(0, 1).equals("/")) {
        tmpUrl = tmpUrl.substring(1, tmpUrl.length());
      }
    } else { // last
      while (tmpUrl.substring(tmpUrl.length() - 1).equals("/")) {
        tmpUrl = tmpUrl.substring(0, tmpUrl.length() - 1);
      }
    }
```

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

```java
    return tmpUrl;
  }
}
```

sep 14, 17 5:57 **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;

import org.apache.log4j.Logger;

import ar.fiuba.taller.loadTestConsole.UsersController;

public class HttpRequester {

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

  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);
      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;
```

sep 14, 17 5:57 **HttpRequester.java** Page 2/3

```java
      StringBuffer response = new StringBuffer();
      while ((inputLine = in.readLine()) ≠ null) {
        response.append(inputLine);
      }
      in.close();
      if ("".equals(response.toString())) {
        return null;
      } else {
        return response.toString();
      }
  }

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

  // 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();
      // add reuqest method
      con.setRequestMethod("PUT");
      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
```

sep 14, 17 5:57 **HttpRequester.java** Page 3/3

```
133         con.setDoOutput(true);
134         DataOutputStream wr = new DataOutputStream(con.getOutputStream());
135         wr.writeBytes(data);
136         wr.flush();
137         wr.close();
138 
139         BufferedReader in = new BufferedReader(
140             new InputStreamReader(con.getInputStream()));
141         String inputLine;
142         StringBuffer response = new StringBuffer();
143 
144         while ((inputLine = in.readLine()) ≠ null) {
145           response.append(inputLine);
146         }
147         in.close();
148 
149         // print result
150         return response.toString();
151     }
152 
153 }
```

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;
  }

}
```

sep 12, 17 19:39 **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.CancellationException;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.Semaphore;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;
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);
    Semaphore noUsersSem = new Semaphore(0);

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

    logger.info("Iniciando UsersController");

    if (it.hasNext()) {
```

```
67        pair = it.next();
68        deltaUsers = pair.getValue() - totalUsersCount;
69        oldTime = pair.getKey();
70      }
71
72      try {
73        while (¬Thread.interrupted()) {
74          logger.info(
75              "Usuarios corriendo en el pool: " + totalUsersCount);
76          logger.info("Usuarios que se deben ingresar al pool: "
77              + deltaUsers);
78          totalUsersCount += updateUsers(totalUsersCount, deltaUsers,
79              executorService);
80          logger.info("Usuarios ingresados");
81          logger.info("Usuarios totales: " + totalUsersCount);
82          summaryQueue.put(new UserStat(totalUsersCount));
83          if (it.hasNext()) {
84            pair = it.next();
85            if (pair.getKey() > oldTime) {
86              sleepTime = pair.getKey() - oldTime;
87            }
88            deltaUsers = pair.getValue() - totalUsersCount;
89            oldTime = pair.getKey();
90            patternTime.set(oldTime);
91            logger.debug("Valores para la proxima corrida: "
92                + pair.getKey() + "-" + pair.getValue());
93          } else {
94            logger.info(
95                "No hay mas usuarios para agregar. Me quedo bloqueado en el semaforo.");
96            noUsersSem.acquire();
97          }
98          logger.info("Tiempo a dormir hasta el proximo pulso: "
99              + sleepTime);
100         Thread.sleep(sleepTime * Constants.SLEEP_UNIT);
101       }
102     } catch (InterruptedException e) {
103       logger.info(
104           "Senial de interrupcion recibida. Eliminado los Usuarios.");
105     } finally {
106       executorService.shutdownNow();
107     }
108   }
109
110   /*
111    * Actualiza el pool de threads con los usuarios pasados Retorna la cantidad
112    * de usuarios agregados o eliminados
113    */
114   @SuppressWarnings("unchecked")
115   private int updateUsers(int totalUsersCount, int deltaUsers,
116       ExecutorService executorService) {
117     int usersToAdd = 0;
118
119     if (totalUsersCount + deltaUsers ≥ maxUsers) {
120       usersToAdd = maxUsers - totalUsersCount;
121     } else if (totalUsersCount + deltaUsers < 0) {
122       usersToAdd = 0;
123     } else {
124       usersToAdd = deltaUsers;
125     }
126
127     logger.info("Usuarios a agregar: " + usersToAdd);
128     if (usersToAdd > 0) {
129       logger.info("Agregando usuarios");
130       // Disparo los users
131       for (int i = 0; i < usersToAdd; i++) {
132         futures.add((Future<User>) executorService.submit(
```



```
133             new User(propertiesMap, summaryQueue, reportQueue, i)));
134       }
135
136     } else if (usersToAdd < 0) {
137       logger.info("Eliminando " + usersToAdd      + " usuarios");
138       int tmpUsersToAdd = Math.abs(usersToAdd);
139       Iterator<Future<User>> it = futures.iterator();
140       Future<User> f;
141       int retry;
142
143       for (int i = 0; i < tmpUsersToAdd; i++) {
144         logger.debug("Matando al usuario -> " + i);
145         retry = 0;
146         f = it.next();
147 //        try {
148 //          while(retry < Constants.USER_KILL_RETRY) {
149             logger.info("Cancelando usuario. Intento nro: " + (retry));
150             f.cancel(true);
151             logger.debug("AAAA");
152 //            try {
153 //              logger.debug("bbbbbbbb");
154 //              f.get(Constants.USERS_TIMEOUT, TimeUnit.MILLISECONDS);
155 //              logger.debug("Usuario " + i + "Cancelado");
156 //            } catch (TimeoutException e) {
157 //              logger.error("Timeout expirado");
158 //              if(retry == Constants.USER_KILL_RETRY - 1) {
159 //                try {
160 //                  reportQueue.put(REPORT_EVENT.SCRIPT_EXECUTED);
161 //                } catch (InterruptedException e1) {
162 //                  logger.error("Error al informar terminacion de ejecucion de
    script");
163 //                  logger.debug(e1);
164 //                }
165 //              }
166 //            } finally {
167 //              retry++;
168 //            }
169 //          }
170           it.remove();
171           logger.debug("Usuario cancelado");
172 //        } catch (ExecutionException | InterruptedException e) {
173 //          try {
174 //            logger.debug("Error en la eliminacion de usuarios. Mando el succes
    s");
175 //            reportQueue.put(REPORT_EVENT.SCRIPT_EXECUTED);
176 //          } catch (InterruptedException e1) {
177 //            logger.error("Error al informar terminacion de ejecucion de script
    ");
178 //            logger.debug(e1);
179 //          }
180 //          logger.error("Controlador interrumpido");
181 //          logger.debug(e);
182 //        } catch ( CancellationException e) {
183 //          logger.debug("Cancellation exception");
184 //          logger.debug(e);
185 //          e.printStackTrace();
186 //        }
187 //      }
188       logger.debug("Usuarios cancelados");
189       }
190     }
191   return usersToAdd;
192   }
193 }
```

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

import java.io.FileReader;
import java.io.IOException;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

import org.apache.log4j.Logger;
import org.apache.log4j.MDC;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;

import ar.fiuba.taller.loadTestConsole.Constants.REPORT_EVENT;
import ar.fiuba.taller.utils.HttpRequester;
import ar.fiuba.taller.utils.PageAnalyzer;

public class User implements Runnable {
  final static Logger logger = Logger.getLogger(User.class);
  private Map<String, String> propertiesMap;
  private ArrayBlockingQueue<SummaryStat> summaryQueue;
  private ArrayBlockingQueue<REPORT_EVENT> reportQueue;
  private int id;

  public User(Map<String, String> propertiesMap,
      ArrayBlockingQueue<SummaryStat> summaryQueue,
      ArrayBlockingQueue<REPORT_EVENT> reportQueue, int id) {
    MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
    this.summaryQueue = summaryQueue;
    this.reportQueue = reportQueue;
    this.propertiesMap = propertiesMap;
    this.id = id;
  }

  @SuppressWarnings("unchecked")
  @Override
  public void run() {
    long time_end, time_start, avgTime = 0, successResponse = 0,
        failedResponse = 0;
    String response = null;
    Map<String, String> resourceMap = null;
    Set<Callable<Downloader>> downloadersSet = new HashSet<Callable<Downloader>>
();
    ExecutorService executorService = Executors.newFixedThreadPool(
        Integer.parseInt(propertiesMap.get(Constants.MAX_DOWNLOADERS)));
    JSONParser parser = new JSONParser();
    HttpRequester httpRequester = new HttpRequester();
    Object objScript = null;
    JSONObject objStep = null;
    JSONArray stepsArray = null;
    List<Future<Downloader>> futures = null;
    PageAnalyzer pageAnalyzer = new PageAnalyzer();
    Iterator<JSONObject> it = null;

    logger.info("Iniciando usuario");
    try {
```



```java
      reportQueue.put(REPORT_EVENT.SCRIPT_EXECUTING);
      objScript = parser.parse(
          new FileReader(propertiesMap.get(Constants.SCRIPT_FILE)));
      stepsArray = (JSONArray) ((JSONObject) objScript).get("steps");
      it = stepsArray.iterator();

      while (¬Thread.interrupted()) {
        logger.debug("iddddd: " + id);
        if (it ≡ null ∨ ¬it.hasNext()) {
          avgTime = 0;
          successResponse = 0;
          failedResponse = 0;
          downloadersSet.clear();
          it = stepsArray.iterator();
        }
        objStep = it.next();
        logger.info("Siguiente url a analizar: "
            + (String) objStep.get("url"));
        logger.info("Metodo: " + (String) objStep.get("method"));
        logger.info("headers: " + (String) objStep.get("headers"));
        logger.info("Body: " + (String) objStep.get("body"));
        time_start = System.currentTimeMillis();
//        try {
        response = httpRequester.doHttpRequest(
            (String) objStep.get("method"),
            (String) objStep.get("url"),
            (String) objStep.get("headers"),
            (String) objStep.get("body"),
            Integer.parseInt(
                propertiesMap.get(Constants.HTTP_TIMEOUT))
                * Constants.SLEEP_UNIT);
//        } catch (IOException e) {
//          logger.error("Error en el http -->" + id);
//          logger.debug(e);
//        }
//        response = "http://www.fi.uba.ar";
        logger.debug("Request listo");
        time_end = System.currentTimeMillis();
        avgTime = time_end - time_start;
        if (response ≡ null) {
          failedResponse++;
        } else {
          successResponse++;
          resourceMap = pageAnalyzer.getResources(response,
              (String) objStep.get("url"));
          reportQueue.put(REPORT_EVENT.URL_ANALYZED);
          for (Map.Entry<String, String> entry : resourceMap
              .entrySet()) {
            logger.debug("tipo: " + entry.getKey());
            logger.debug("recurso: " + entry.getValue());
            downloadersSet.add(new Downloader(reportQueue,
                entry.getValue(), entry.getKey(), propertiesMap,
                summaryQueue));
          }
          futures = executorService.invokeAll(downloadersSet);
          logger.info("Esperando Downloaders");
          for (Future<Downloader> future : futures) {
            future.get();
          }
        }
        summaryQueue.put(new RequestStat(successResponse,
            failedResponse, avgTime));
      }
    } catch (Exception e) {
      logger.error("Error en la ejecucion del user --> " + id);
      logger.debug(e);
```

sep 12, 17 20:01 **User.java** Page 3/3

```
132       } finally {
133         logger.debug("ME MUEROOOO -> " + id);
134         try {
135           logger.info("User cancelado. Eliminando downloaders.");
136           executorService.shutdownNow();
137           reportQueue.put(REPORT_EVENT.SCRIPT_EXECUTED);
138         } catch (InterruptedException e1) {
139           logger.error("Error al enviar estadisticas a la cola de reportes -->" + id);
140           logger.debug(e1);
141         }
142       }
143     }
144   }
```

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 18:45 **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)/2);
      } catch (InterruptedException e) {
        logger.info("SummaryPrinter interrumpido");
      }
    }
    logger.info("SummaryPrinter finalizado");
  }
}
```

sep 10, 17 19:16 **Summary.java** Page 1/1

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

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

  public Summary() {
    avgDownloadTime = 0;
    successfullrequest = 0;
    failedrequest = 0;
    users = 0;
    requestCounter = 0;
    timeCounter = 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) {
    requestCounter++;
    timeCounter += time;
    avgDownloadTime = timeCounter / requestCounter;
  }

  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 18:45 **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)/2);
      } 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  }
```

sep 10, 17 20:14 **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  {
  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;
  }

  public void start() {
    // 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);
  }
}
```

sep 10, 17 23:02 **LoadTestConsole.java** Page 1/3

```
1  package ar.fiuba.taller.loadTestConsole;
2
3  import ar.fiuba.taller.loadTestConsole.Constants.REPORT_EVENT;
4  import java.io.File;
5  import java.io.FileNotFoundException;
6  import java.io.IOException;
7  import java.util.HashMap;
8  import java.util.InputMismatchException;
9  import java.util.Map;
10 import java.util.Properties;
11 import java.util.Scanner;
12 import java.util.TreeMap;
13 import java.util.concurrent.ArrayBlockingQueue;
14 import java.util.concurrent.Semaphore;
15 import java.util.concurrent.atomic.AtomicInteger;
16 import java.util.regex.Pattern;
17
18 import org.apache.log4j.Logger;
19 import org.apache.log4j.MDC;
20
21 public class LoadTestConsole {
22   private Map<String, String> propertiesMap;
23   private Map<Integer, Integer> usersPatternMap;
24
25   final static Logger logger = Logger.getLogger(LoadTestConsole.class);
26
27   public LoadTestConsole() throws IOException {
28     MDC.put("PID", String.valueOf(Thread.currentThread().getId()));
29     // Cargo la configuracion del archivo de properties
30     loadProperties();
31   }
32
33   public void start() throws FileNotFoundException {
34     Semaphore fileChangeSem = new Semaphore(0);
35     ArrayBlockingQueue<SummaryStat> summaryQueue =
36         new ArrayBlockingQueue<SummaryStat>(
37         Integer.parseInt(
38             propertiesMap.get(Constants.SUMMARY_QUEUE_SIZE)));
39     ArrayBlockingQueue<REPORT_EVENT> reportQueue =
40         new ArrayBlockingQueue<REPORT_EVENT>(
41         Integer.parseInt(
42             propertiesMap.get(Constants.REPORT_QUEUE_SIZE)));
43     Summary summary = new Summary();
44     Report report = new Report();
45     AtomicInteger patternTime = new AtomicInteger(0);
46
47     // Creo los threads
48     Thread usersControllerThread = null;
49     PatternFileWatcher pfw = new PatternFileWatcher(
50         propertiesMap.get(Constants.USERS_PATTERN_FILE), fileChangeSem,
51         Integer.parseInt(
52             propertiesMap.get(Constants.FILE_WATCHER_TIMEOUT)));
53     Thread summaryControllerThread = new Thread(
54         new SummaryController(summaryQueue, summary));
55     Thread summaryPrinterThread = new Thread(
56         new SummaryPrinter(summary, propertiesMap));
57     Thread reportControllerThread = new Thread(
58         new ReportController(reportQueue, report));
59     Thread reportPrinterThread = new Thread(
60         new ReportPrinter(report, propertiesMap));
61
62     logger.info("Iniciando LoadTestConsole");
63     logger.info("Iniciando los threads");
64     //patternFileWatcherThread.start();
65     pfw.start();
66     summaryControllerThread.start();
```

sep 10, 17 23:02 **LoadTestConsole.java** Page 2/3

```
67     summaryPrinterThread.start();
68     reportControllerThread.start();
69     reportPrinterThread.start();
70
71     while (¬Thread.interrupted()) {
72       logger.info("Cargando patron de usuarios");
73       loadUserPattern(patternTime);
74       logger.info("Disparando el usersControllerThread");
75       usersControllerThread = new Thread(
76           new UsersController(propertiesMap, usersPatternMap,
77               patternTime, summaryQueue, reportQueue));
78       usersControllerThread.start();
79       try {
80         // Me quedo esperando hasta que cambie el archivo
81         logger.info("Esperando hasta que cambie el archivo");
82         fileChangeSem.acquire();
83       } catch (InterruptedException e) {
84         // Do nothing
85         logger.error("No se ha podido tomar el semaforo");
86       }
87       logger.info(
88           "Cambio el archivo. Interrumpiendo el usersControllerThread");
89       System.out.println(
90           "Cambio el archvio. Recargando patron de usuarios...");
91       usersControllerThread.interrupt();
92       try {
93         usersControllerThread.join();
94       } catch (InterruptedException e) {
95         logger.error("Error al joinear el usersControllerThread");
96         logger.debug(e);
97       }
98     }
99     logger.info("Interrumpiendo los threads de estadisticas");
100    summaryControllerThread.interrupt();
101    summaryPrinterThread.interrupt();
102    reportControllerThread.interrupt();
103    reportPrinterThread.interrupt();
104    logger.info("Joineando los threads");
105    try {
106      summaryControllerThread.join();
107      summaryPrinterThread.join();
108      reportControllerThread.join();
109      reportPrinterThread.join();
110    } catch (InterruptedException e) {
111      logger.error("Error al joinear los threads de reportes");
112      logger.debug(e);
113    }
114  }
115
116  private void loadProperties() throws IOException {
117    logger.info("Cargando configuracion");
118    propertiesMap = new HashMap<String, String>();
119    Properties properties = new Properties();
120    try {
121      properties.load(Thread.currentThread().getContextClassLoader()
122          .getResourceAsStream(Constants.PROPERTIES_FILE));
123    } catch (IOException e) {
124      System.err.println(
125          "No ha sido posible cargar el archivo de propiedades");
126      throw new IOException();
127    }
128    for (String key : properties.stringPropertyNames()) {
129      String value = properties.getProperty(key);
130      propertiesMap.put(key, value);
131      logger.debug("Parametro cargado: " + key + "->" + value);
132    }
```

sep 10, 17 23:02 **LoadTestConsole.java** Page 3/3

```
133   }
134
135   private void loadUserPattern(AtomicInteger patternTime)
136       throws FileNotFoundException {
137     File file = new File(propertiesMap.get(Constants.USERS_PATTERN_FILE));
138     int nextIntTime, nextIntUser;
139     if (file ≡ null ∨ ¬file.canRead()) {
140       System.out.println(
141           "No se ha podido abrir el archivo con el patron de usuarios.");
142     }
143     usersPatternMap = new TreeMap<Integer, Integer>();
144     @SuppressWarnings("resource")
145     final Scanner s = new Scanner(file)
146         .useDelimiter(Pattern.compile("(\\n)|:"));
147     try {
148       while (s.hasNext()) {
149         nextIntTime = s.nextInt();
150         nextIntUser = s.nextInt();
151         logger.debug(
152             "Tiempo desde el cual cargar: " + patternTime.get());
153         if (nextIntTime ≥ patternTime.get()) {
154           usersPatternMap.put(nextIntTime, nextIntUser);
155           logger.debug("Patron cargado: " + nextIntTime + ","
156               + nextIntUser);
157         }
158       }
159     } catch (InputMismatchException e) {
160       logger.error("Error al cargar el archivo de patrones de usuario");
161       logger.debug(e);
162     }
163   }
164 }
```

sep 10, 17 20:33 **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     boolean statSend = false;
39
40     logger.info("Iniciando Downloader.");
41     logger.info("Url a descargar: " + url);
42     logger.info("Tipo de recurso: " + type);
43     try {
44       reportQueue.put(REPORT_EVENT.RESOURCE_DOWNLOAD);
45       statSend = true;
46       time_start = System.currentTimeMillis();
47       response = httpRequester.doHttpRequest("get", url, null, null,
48           Integer.parseInt(
49               propertiesMap.get(Constants.HTTP_TIMEOUT)));
50       time_end = System.currentTimeMillis();
51       time_elapsed = time_end − time_start;
52       reportQueue.put(Constants.TYPE_RESOURCE_MAP.get(type));
53     } catch (Exception e) {
54       // Do nothing
55     } finally {
56       try {
57         if (response ≡ null) {
58           failedResponse++;
59         } else {
60           successResponse++;
61         }
62         summaryQueue.put(new RequestStat(successResponse,
63             failedResponse, time_elapsed));
64         if(statSend) {
65           reportQueue.put(REPORT_EVENT.RESOURCE_DOWNLOADED);
66         }
```

sep 10, 17 20:33 **Downloader.java** Page 2/2

```
67          } catch (InterruptedException e) {
68            // Do nothing
69          }
70        }
71        logger.info("Downloader terminado.");
72        return null;
73      }
74    }
```

sep 12, 17 18:49 **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    public static final long USERS_TIMEOUT = 100;
24    public static final int USER_KILL_RETRY = 10;
25
26    public static final Map<String, String> RESOURCE_MAP;
27    static {
28      RESOURCE_MAP = new HashMap<String, String>();
29      RESOURCE_MAP.put("LINK", "href");
30      RESOURCE_MAP.put("SCRIPT", "src");
31      RESOURCE_MAP.put("IMG", "src");
32    }
33
34    public static enum REPORT_EVENT {
35      URL_ANALYZED, SCRIPT_DOWNLOADED, LINK_DOWNLOADED, IMG_DOWNLOADED,
36      SCRIPT_EXECUTING, SCRIPT_EXECUTED, RESOURCE_DOWNLOAD,
37      RESOURCE_DOWNLOADED
38    };
39
40    public static final Map<String, REPORT_EVENT> TYPE_RESOURCE_MAP;
41    static {
42      TYPE_RESOURCE_MAP = new HashMap<String, REPORT_EVENT>();
43      TYPE_RESOURCE_MAP.put("LINK", REPORT_EVENT.LINK_DOWNLOADED);
44      TYPE_RESOURCE_MAP.put("SCRIPT", REPORT_EVENT.SCRIPT_DOWNLOADED);
45      TYPE_RESOURCE_MAP.put("IMG", REPORT_EVENT.IMG_DOWNLOADED);
46    }
47  }
```

sep 14, 17 7:07 **Table of Content** Page 1/1

Table of Contents