Eastern Academic Journal

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

Supporting materials to article entitled
APPROACH FOR DESIGN PATTERN' APPLICATION IN THE DEVELOPMENT OF INFORMATION SYSTEMS

Mariya Armyanova

University of Economics, Varna, Bulgaria

The article could be downloaded from here

CampaignSvc.java еквивалент на process

```
@Path("/campaign/") //process реализира връзката с потребителския интерфейс
public class CampaignSvc {
   @POST
   @Path("start/")
   public void startCampaign(
         @PathParam("campaign") String campaignId,
         @PathParam("minLists") Integer numberLists,
         @PathParam("startDate") Integer lastSpammed) {
      final String ACCUM_INIT_URL = "/accumulator/init/";
      String accUrl = ACCUM_INIT_URL + campaignId + "/3/" + numberLists;
      String accumSession = RestCaller.callPostString(accUrl, null);

      final String[] ANALIZE_URLS = {
         "/analizes/reviewproduct/",
         "/analizes/buyproduct/",
         "/analizes/amount/"};
      for (String url : ANALIZE_URLS ){
         RestCaller.callPostAsync(url+accumSession, null);
      }   }}
```

За извикване на услугата RestCaller.java: за обръщение към сървисите в случая синхронно и асинхронно обръщение

```
public class RestCaller {
   private static void prepareAndSend(byte[] postData, HttpURLConnection conn)
         throws ProtocolException, IOException {
      conn.setRequestMethod("POST");
      conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
      conn.setRequestProperty("charset", "utf-8");
      conn.setDoOutput(true);
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
    if (postData != null){
        conn.setRequestProperty("Content-Length", Integer.toString(postData.length));
        conn.getOutputStream().write(postData);
        conn.getOutputStream().flush();
    }else{
        conn.setRequestProperty("Content-Length", "0");
    }    }
public static void callPostNoResult(String urlAddr, byte[] postData) {
    try {
        URL url = new URL(urlAddr);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        prepareAndSend(postData, conn);
        conn.disconnect();
    } catch (Exception ex) {
    }    }
public static String callPostString(String urlAddr, byte[] postData) {
    StringBuilder output = new StringBuilder();
    try {
        URL url = new URL(urlAddr);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        prepareAndSend(postData, conn);
        BufferedReader br = new BufferedReader(new InputStreamReader(
              (conn.getInputStream())));
        String line;
        while ((line = br.readLine()) != null) {
           output.append(line);
        }
        conn.disconnect();
    } catch (Exception ex) {
    }
    return output.toString();
}
private static class AsyncCaller implements Runnable {
    private final String urlAddr;
    private final byte[] postData;
    public AsyncCaller(String urlAddr, byte[] postData) {
        this.urlAddr = urlAddr;
        this.postData = postData.clone();
    }
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
    @Override
    public void run() {
        RestCaller.callPostNoResult(urlAddr, postData);
    }
}

public static void callPostAsync(String urlAddr, byte[] postData) {
    Thread thread;
    thread = new Thread(new AsyncCaller(urlAddr, postData));
    thread.setDaemon(true);
    thread.start();
}
public static String stringListToParamList(List<String> list, String paramName) {
    StringBuilder builder = new StringBuilder();
    list.forEach((elem) -> {
        builder.append(paramName).append("=").append(elem).append("&");
    });

    if (builder.length() > 0) {
        builder.deleteCharAt(builder.length() - 1);
    }
    return builder.toString();
}}
```

Акумулатор:

```
@Path("/accumulator/")
public class UserListAccumulatorSvc {

    //Може да се преместят в конфигурационен файл (xml), за да се позволи лесна промяна на адресите на услугите без прекомпилация на кода.
    static final String RESULT_URL = "/mailSender/";
    SessionIdGenrator generator;
    final Map<String, AccumulatorSession> map;
    SessionCleanerTask task;
    public UserListAccumulatorSvc() {
        generator = new SessionIdGenrator();// създава нов генератор на сесийни ключове
        map = new HashMap<>(); //създава съответствие между сесията и идентификатора ѝ
        task = new SessionCleanerTask(this); // пуска се нишката, която изчиства незавършели сесии
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
    task.start();
  }

  public Map<String, AccumulatorSession> getMap() {
    return map;
  }

  @POST //инициализира акумулатора
  @Path("init/{varCampaign}/{varNumbers}/{varMinToPass}")
  @Produces(MediaType.APPLICATION_JSON) // типично за REST връща резултата като JSON.
  public synchronized String startAccumulatorSession(
      @PathParam("varCampaign") String campaign, //параметрите се извличат от URL
      @PathParam("varNumbers") Integer numberLists,
      @PathParam("varMinToPass") Integer minToPass) {

    String sessionId = generator.getNextId();
    AccumulatorSession session = new AccumulatorSession(
        numberLists, minToPass, campaign);
    map.put(sessionId, session);
    return sessionId;
  }

  @POST
  @Path("data/{varID}") // методът натрупва данните към съответната сесия на Accumulator-a.
  @Produces(MediaType.APPLICATION_JSON)
  public Response addAccumulatorData(
      @PathParam("varID") String sessionID,
      @PathParam("users") List<String> users) {
    AccumulatorSession session;

    synchronized (map) {
      session = map.get(sessionID);
    }
    if (session == null) {
      return Response.status(Response.Status.NOT_FOUND) //невалидна сесия
          .entity("Session not found").build();
    }

    synchronized (session) { //точката на синхронизация на анализите, които връщат данни
      users.forEach((user) -> {
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
            int count = 1;
            Integer num = session.users.get(user);
            if (num != null) {
                count = num + 1;
            }
            session.users.put(user, count);
        });

        session.setProcessed(); //маркира, че сесията е обработвана
        if (session.numToWait == 0) {
            processFinalList(session);
            synchronized (map) {
                map.remove(sessionID);
            }
        }
    }
    return Response.ok().build();
}

public void processFinalList(AccumulatorSession session) { //изпраща резултата към втората стъпка
    List<String> usersToNotify = new LinkedList<>();
    for (Map.Entry<String, Integer> entry : session.users.entrySet()) {
        if (entry.getValue() >= session.minLists) {
            usersToNotify.add(entry.getKey());
        }       }

    String url = RESULT_URL + session.campaignId + "/"; //сголбява адреса на втората стъпка
    byte[] postData = RestCaller.stringListToParamList(usersToNotify, "userId")
            .getBytes(StandardCharsets.UTF_8);
    RestCaller.callPostNoResult(url, postData);
}

}
```

Изтрива незавършените сесии:

```
public class SessionCleanerTask extends Thread {
    private boolean running = true;
    private final static int POLL_INTERVAL = 5000; // 5 sec.
    private final static int PROCESSING_TIMEOUT = 10000; // 10 sec.
    private final UserListAccumulatorSvc accumulator;
    public SessionCleanerTask(UserListAccumulatorSvc accum) {
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
        this.accumulator = accum;
    }

    @Override
    public void run() {
        while (running) {
            try {
                sleep(POLL_INTERVAL); // колко често ги проверява
                List<Map.Entry<String, AccumulatorSession>> toRemove = new LinkedList<>(); //прави списък от сесиите за премахване
                for (Map.Entry<String, AccumulatorSession> entry
                        : accumulator.getMap().entrySet()) {
                    if (entry.getValue().timeIdle() > PROCESSING_TIMEOUT) { // отбелязва за премахване не обработваните сесии
                        toRemove.add(entry);
                    }
                }
                synchronized (accumulator.getMap()) { //синхронизира се, за да не се наруши структурата на таблицата
                    toRemove.forEach((s) -> {
                        accumulator.getMap().remove(s.getKey());
                        accumulator.processFinalList(s.getValue());
                    });
                }
            } catch (InterruptedException ex) {
                return;
            }      }    }
```

Самата сесия като клас

```
public class AccumulatorSession implements Serializable {
    private final Calendar calendar; //клас с функции за време, включително текущото
    private Date lastUsed;
    String campaignId;
    int numToWait;
    int minLists;
    Map<String, Integer> users;
    public AccumulatorSession(int numTasksToWait, int minListsToPass, String campaignID) {
        calendar = Calendar.getInstance();
        lastUsed = calendar.getTime();
        numToWait = numTasksToWait;
        minLists = minListsToPass;
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
        users = new HashMap<>();
        campaignId = campaignID;
    }

    void setProcessed() {
        --numToWait; //намалява броя на чакащите
        lastUsed = calendar.getTime(); // и се отбелязва кога е обработвана сесията
    }
    long timeIdle() { //колко време не е ползвана. Капсулира се в сесията работата с времето.
        return calendar.getTime().getTime() - lastUsed.getTime();
    }
```

Анализите са

```
@Path("/analizes/")
public class ReviewAnalyzeSvc {
    static final String RESULT_URL = "/accumulator/data/";
    @POST
    @Path("reviewproduct/{varCampaignID}") //Подава се кампанията
    public void doAnalyze(@PathParam("varCampaignID") String campaignID,
            @PathParam("varAccSessionID") String accSessionId) {//подава се сесията на акумулатора
        List<String> users = selectAllUsers();
//списъка от потребители се обръща в данни подходящи за параметри на REST услугата
        byte[] postData =  RestCaller.stringListToParamList(users, "users")
                .getBytes(StandardCharsets.UTF_8);
        String url = RESULT_URL + accSessionId + "/";
        RestCaller.callPostNoResult(url, postData); //сглобява URL и извиква accumulator
    }
    private List<String> selectAllUsers() {
// Код за анализ и връща потребителите, които отговарят на критериите
        return users;
    }
}
```

За втората стъпка. Класът, отговарящ за попълване на данните от кампанията и изпращане на писмата:

```
public class CampaignMail {
    private final String SMTP_SRV_ADDRES = "smtp.myaddres.com"; //адреса на SMTP сървъра
    private final String SENDER_ADDRES = "sails@myaddress.com"; //адреса, от който се праща
    private Session session;
    private String subject; //subject на писмото
    private String text; //текст на писмото
    public void fillData(User user) {
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
    //Попълват се данните на клиента в текста на писмото
  }
  public CampaignMail(String campaignID) {
    Properties properties = System.getProperties();
    properties.setProperty("mail.smtp.host", SMTP_SRV_ADDRES);
    session = Session.getDefaultInstance(properties);
    //попълва се шаблона на писмото и subject от данните на кампанията
  }
  public void send(String address) throws Exception { //изпраща писмото
    MimeMessage message = new MimeMessage(session);
    message.setFrom(new InternetAddress(SENDER_ADDRES));
    message.addRecipient(Message.RecipientType.TO, new InternetAddress(address));
    message.setSubject(subject);
    message.setText(text);
    Transport.send(message);
  }}
```

Той изпраща асинхронно писмата, за да се върне максимално бързо управлението. Изпращането на писмата може да е бавно и не е добре да се държи конекцията (връзката) между сървиса и клиента.

```
public class MailSenderTask extends Thread {
  List<String> userIds;
  CampaignMail mail;
  public MailSenderTask(List<String> users, CampaignMail mail) {
    userIds = users;
    this.mail = mail;
  }
  public void run() {
    for (String id : userIds) {
      User user = UserDao.getInstance().getUserById(id);
      mail.fillData(user); //попълва данните на потребителя
      try {
        mail.send(user.getEmail()); //изпраща писмото
      } catch (Exception ex) {
      }     }   }}
```

Кодът на самата услуга:

```
@Path("/mailSender/")
public class MailSenderSvc {
  @POST
  @Path("send/{varCampaignID}")
  public Response addAccumulatorData(
      @PathParam("varCampaignID") String campaignID,
```

ISSN: 2367-7384
Issue 4, pp. 62-75, December, 2017

```
        @PathParam("users") List<String> users) {
        CampaignMail mail = new CampaignMail(campaignID); //създава обект mail
        MailSenderTask task = new MailSenderTask(users, mail); //създава задача (нишка) за изпращане на
```

псимата стартира я, откачава се от нея и я оставя да си работи.

```
        task.start();
        task.setDaemon(true);
        return Response.ok().build();
    }}
//Шаблонът singleton
public class UserDao {
    private static UserDao theDao;
    private UserDao(){}
    public static synchronized UserDao getInstance(){
        if (theDao == null){
            theDao = new UserDao();
        }
        return theDao;
    }
    public User getUserById(String Id){
        //зарежда данните за потребителя от базата от данни по ID
    }}
```