**Problems**

Xử lý log trong dự án OCS Gateway là một nhiệm vụ quan trọng, có rất nhiều các tác vụ phía sau cần phải sử dụng (tra cứu lịch sử giao dịch, Reporting, CDA …)

Ngoài ra, trong hệ thống OCS Gateway có rất nhiều các module thành phần (có khoảng 20 service dịch vụ), mỗi module này lại có định dạng ghi log riêng, nhiều services lại có nhiều hơn một instances khi triển khai.

Trong tương lai hoàn toàn có thể sẽ phát sinh việc xuất hiện một định dạng file log mới. Công việc khi xử lý file log này phải hết sức đơn giản, tốn ít công sức nhất có thể, không được gây ra side-effect đến những phần đã có (đảm bảo tính đúng đắn của các phần đã có)

**Methodologies**

Hệ thống xử lý log cần phải đảm bảo được các tính chất sau (các tính chất trong OOD):

* Đóng gói được các phần sử dụng chung
* Tính đóng mở (không cho phép sửa đổi những phần đã có, welcome các phần mở rộng - ví dụ như bổ sung thêm 1 vài định dạng log file mới)

Tạo ra một loạt các class có vai trò gần như tương tự nhau, mỗi class xử lý 1 loại file log (áp dụng Strategy Pattern), để switch giữa các file log cần xử lý khi chương trình chạy runtime, sử dụng Simple Factory Method.

Như vậy, về mặt phương pháp ta sẽ sử dụng 2 pattern sau đây:

* Simple Factory Method (thực chất đây không phải là một Patterns, nó nằm trong nhóm Factory Pattern)
* Strategy Patterns

Cụ thể hơn, về giải pháp tham khảo phần Solution

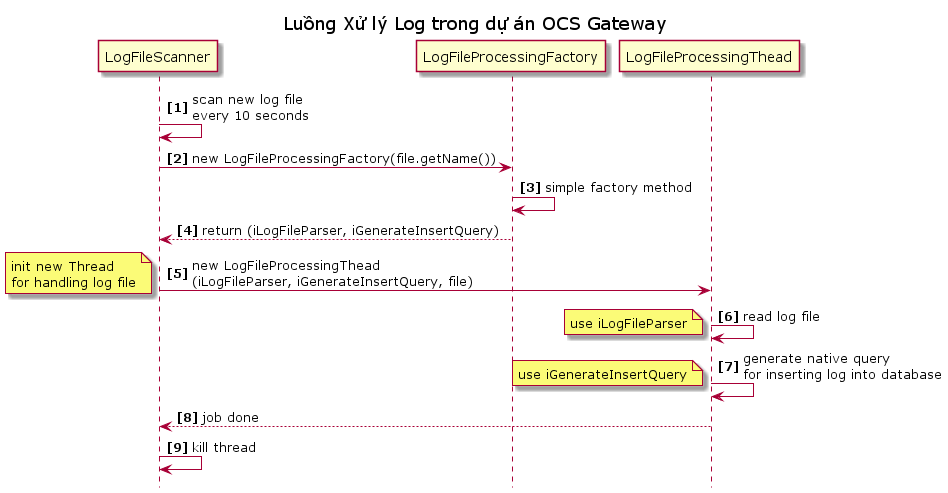
**Solutions**

Trước hết cần quan sát một chút về quy trình xử lý log, việc này rất quan trọng trong kỹ thuật lập trình, để có thể viết nên một chương trình hạn chế được sự thay đổi của yêu cầu trong tương lai thì phải nhìn rõ vấn đề này. Sẽ phải chấp nhận rằng dù chương trình và yêu cầu sẽ thay đổi nhưng có những logic nghiệp vụ (luồng xử lý chính) không bao giờ được thay đổi. Trong trường hợp bài toán của chúng ta đang đề cập đến chính là luồng xử lý file log.

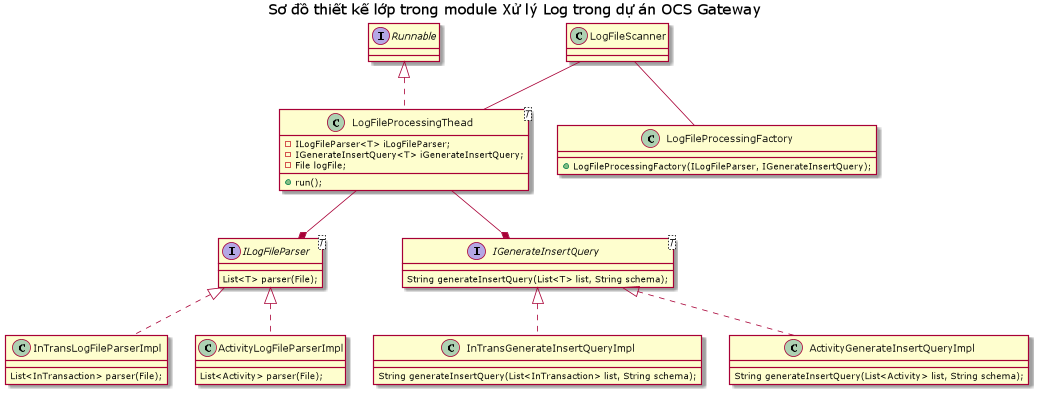
Luồng xử lý file log này bao gồm các bước chính sau đây:

* Quét folder chứa file log (toàn bộ các services khác khi tạo ra file log sẽ được tập trung về một nơi - việc quy tập toàn bộ các file này về một folder là công việc của dịch vụ ftp trong hệ thống OCS Gateway)
* Đọc nội dung file log (mỗi dòng log chứa một transaction log, các dòng log này có cấu trúc, hai cấu trúc đang được sử dụng là JSON hoặc các element split với nhau bằng các kí tự đặc biệt - ví dụ ###)
* Chuyển đổi nội dung dòng log thành các Object (mỗi một loại log sẽ có một class mô tả cấu trúc của log này)
* Chuyển dữ liệu log (trong các class) thành một câu SQL native (phục vụ cho việc insert vào database)
* Hệ thống xử lý log sẽ thực thi câu SQL native để đưa dữ liệu log vào trong database, phục vụ cho các công việc sau này (dữ liệu cho truy vấn lịch sử giao dịch, dữ liệu thô cho module reporting, CDA …)

Sau khi đã xác định được luồng logic chính (không bao giờ thay đổi) của phần xử lý log, ta quan tâm đến việc mô hình hóa luồng này bằng các thiết kế lớp



*Hình 1:* [*Sơ đồ*](http://www.plantuml.com/plantuml/uml/fLFDIiD04BxdAGPFBQZKgqWlGejG_C4YWlKmoKwIrSsixavQ-nm-WG_WmNFnv9FqJTp4jYgcfU0jcM_t-vl4tsj-G8S5A4gnDDApBATO45oHKyWOYHP3C2ZdisVEu7e-UmNpyGu3cu4u6qPgFdk3ZnU6i-DBwAFG1ATHhXL1Qgt4zYd2KYoNHKmETexYxatdTYUA0a5F6xfCa3a0UrtuFJa47nw0QG8cQAK16J6DoKtXe0EU4ilARw0vTpOXxpLdFKp4kkbXpRKERLK2-ndLuLljzehvpydD6buN3w6lT76s8CcjsaJM4DoHb8wXfPUtq7bokw3x5F3Gy0c7TxaeGndjhIeOveHgJG4rDkAjr7RhFRGZja9WA9K8GBEMcdcOky0K9gbra2Ch4zXNouk85LJt6igeBMmoNX5_C_rKBpt1R-V_4ieMcO5Hz9ZWiKezuYgHhcjOHj8i5XGAnkWR73LqjuMlXc_XpiQWBDCsFyozDWRaQmkV) *thiết kế luồng chính của module xử lý Log*



*Hình 2:* [*Sơ đồ*](http://www.plantuml.com/plantuml/uml/jLFDIiD04BxdAUPORVi45K64baBnhqMyZicqNRgPbDsD4iI748-U-W0Ul5h4GyNto9kuIPfgcaH5y9A4RxvlvljcitecU8uAVV1eZA4qtO3D4Ve4vwGyP7ICC99W-Bw0jxjuUGzcAkBdLmCp-mGPBn_cO5J04_03BxJCYtZv27Br0eDWigvuyV89LWk6uyCXzD3GDKQEuqhKEc5rXQIXYyoaYk292bpIMl1aD2NqzaRx8Fov99_OQ3WBcV7IJhnn0DhGpqMeDAcKMq0w6Qj7Teorq6UBcTEGL9HoA_1CaNG0cNLAa2Qea1ix7UUspcmNNHEeAFNLh2qtYePRbTRICOvWGsgCBbM4J8OCX3R9zpn56mcX9Ap9dSY7HbbRC2ap6dbdQT-jdAZTAVcOJSXMqEUHGjO5StrxKb-FgsXzOtijjJiH0PSZ5FfNP2XDgU3yAkIMZny27rZQbJ3HNnBdshgmUVq_qtvwg0rQl9ZGRjV-oR-XfXTuqtGBjuhToYpVyWlRTJu0) *thiết kế lớp của module xử lý Log*

**Implementation**

Dưới đây là một số đoạn mã được viết trong chương trình. Tham khảo thêm mã nguồn dự án tại đường dẫn sau: [Link](http://git.vnpt-technology.vn/projects/SOG/repos/ocsgw-log-service/browse)

public interface IGenerateInsertQuery<T> {

String generateInsertQuery(List<T> list, String schema);

}

public interface ILogFileParser<T> {

List<T> parser(File file);

}

public class ActivityLogFileParserImpl implements ILogFileParser<Activity> {

private static Logger *logger* = LoggerFactory.*getLogger*(ActivityLogFileParserImpl.class);

@Override

public List<Activity> parser(File file) {

List<Activity> results = new LinkedList<>();

try {

//read log file line by line, and create object

} catch (IOException e) {

*logger*.error(e.getMessage(), e);

}

return results;

}

}

public class ActivityGenerateInsertQueryImpl implements IGenerateInsertQuery<Activity> {

private static final String *HEADER\_COLUMNS* = "SESSION\_ID,EVENT\_ID,STATE,TIME\_REQUEST,TIME\_RESPONSE,ACTION,OBJECT\_TYPE," +

"OBJECT\_ID,OBJECT\_NAME,DESCRIPTION,USERNAME,USER\_TYPE,UPDATED,CREATED,CLIENT\_IP";

@Override

public String generateInsertQuery(List<Activity> list, String schema) {

StringBuilder sb = new StringBuilder();

sb.append("INSERT INTO ").append(schema).append(".ACTIVITY ( ")

.append(*HEADER\_COLUMNS*)

.append(" ) WITH TEMP\_TABLE ( ")

.append(*HEADER\_COLUMNS*)

.append(" ) AS ( ")

.append(toValues(list))

.append(" ) SELECT \* FROM TEMP\_TABLE")

;

return sb.toString();

}

}

public class LogFileProcessingFactory {

private ILogFileParser iLogFileParser = null;

private IGenerateInsertQuery iGenerateInsertQuery = null;

public LogFileProcessingFactory(String fileName) {

if(!fileName.endsWith(Constants.LOG\_FILE\_PATTERN.*EXT\_FILE\_LOG*)) return;

if (fileName.startsWith(Constants.LOG\_FILE\_PATTERN.*I\_TRANS*)) {

iLogFileParser = new InTransLogFileParserImpl();

iGenerateInsertQuery = new InTransGenerateInsertQueryImpl();

} else if (fileName.startsWith(Constants.LOG\_FILE\_PATTERN.*O\_TRANS*)) {

iLogFileParser = new OutTransLogFileParserImpl();

iGenerateInsertQuery = new OutTransGenerateInsertQueryImpl();

} else if (fileName.startsWith(Constants.LOG\_FILE\_PATTERN.*ACTIVITY*)) {

iLogFileParser = new ActivityLogFileParserImpl();

iGenerateInsertQuery = new ActivityGenerateInsertQueryImpl();

} else if (fileName.startsWith(Constants.LOG\_FILE\_PATTERN.*ACTIVITY\_CHANGE\_LIST*)) {

iLogFileParser = new ActivityChangeListLogFileParserImpl();

iGenerateInsertQuery = new ActivityChangeListGenerateInsertQueryImpl();

} else if(fileName.startsWith(Constants.LOG\_FILE\_PATTERN.*O\_NEIF*)) {

iLogFileParser = new NEIFTransactionFileParerImpl();

iGenerateInsertQuery = new NEIFTransactionGenerateInsertQueryImpl();

}

}

public ILogFileParser getILogFileParser() {

return iLogFileParser;

}

public IGenerateInsertQuery getIGenerateInsertQuery() {

return iGenerateInsertQuery;

}

}