package com.twitter.search.earlybird\_root.filters;

import java.util.Set;

import com.google.common.base.Joiner;

import org.apache.thrift.TException;

import org.slf4j.Logger;

import com.twitter.finagle.Service;

import com.twitter.finagle.SimpleFilter;

import com.twitter.search.common.metrics.SearchCounter;

import com.twitter.search.common.util.thrift.ThriftUtils;

import com.twitter.search.earlybird.thrift.EarlybirdResponse;

import com.twitter.search.earlybird.thrift.EarlybirdResponseCode;

import com.twitter.search.earlybird\_root.common.EarlybirdRequestContext;

import com.twitter.util.Future;

import com.twitter.util.FutureEventListener;

/\*\*

\* The general framework for earlybird root to track sensitive results.

\*/

public abstract class SensitiveResultsTrackingFilter

extends SimpleFilter<EarlybirdRequestContext, EarlybirdResponse> {

/\*\*

\* The type name is used to distinguish different kinds of sensitive results in log.

\*/

private final String typeName;

/\*\*

\* The mark is to control whether to log expensive information.

\*/

private final boolean logDetails;

/\*\*

\* Constructor helps distinguish different sensitive content trackers.

\* @param typeName The sensitive content's name (e.g. nullcast)

\* @param logDetails Whether to log details such as serialized requests and responses

\*/

public SensitiveResultsTrackingFilter(final String typeName, boolean logDetails) {

super();

this.typeName = typeName;

this.logDetails = logDetails;

}

/\*\*

\* Get the LOG that the sensitive results can write to.

\*/

protected abstract Logger getLogger();

/\*\*

\* The counter which counts the number of queries with sensitive results.

\*/

protected abstract SearchCounter getSensitiveQueryCounter();

/\*\*

\* The counter which counts the number of sensitive results.

\*/

protected abstract SearchCounter getSensitiveResultsCounter();

/\*\*

\* The method defines how the sensitive results are identified.

\*/

protected abstract Set<Long> getSensitiveResults(

EarlybirdRequestContext requestContext,

EarlybirdResponse earlybirdResponse) throws Exception;

/\*\*

\* Get a set of tweets which should be exclude from the sensitive results set.

\*/

protected abstract Set<Long> getExceptedResults(EarlybirdRequestContext requestContext);

@Override

public final Future<EarlybirdResponse> apply(

final EarlybirdRequestContext requestContext,

Service<EarlybirdRequestContext, EarlybirdResponse> service) {

Future<EarlybirdResponse> response = service.apply(requestContext);

response.addEventListener(new FutureEventListener<EarlybirdResponse>() {

@Override

public void onSuccess(EarlybirdResponse earlybirdResponse) {

try {

if (earlybirdResponse.responseCode == EarlybirdResponseCode.SUCCESS

&& earlybirdResponse.isSetSearchResults()

&& requestContext.getParsedQuery() != null) {

Set<Long> statusIds = getSensitiveResults(requestContext, earlybirdResponse);

Set<Long> exceptedIds = getExceptedResults(requestContext);

statusIds.removeAll(exceptedIds);

if (statusIds.size() > 0) {

getSensitiveQueryCounter().increment();

getSensitiveResultsCounter().add(statusIds.size());

logContent(requestContext, earlybirdResponse, statusIds);

}

}

} catch (Exception e) {

getLogger().error("Caught exception while trying to log sensitive results for query: {}",

requestContext.getParsedQuery().serialize(), e);

}

}

@Override

public void onFailure(Throwable cause) {

}

});

return response;

}

private void logContent(

final EarlybirdRequestContext requestContext,

final EarlybirdResponse earlybirdResponse,

final Set<Long> statusIds) {

if (logDetails) {

String base64Request;

try {

base64Request = ThriftUtils.toBase64EncodedString(requestContext.getRequest());

} catch (TException e) {

base64Request = "Failed to parse base 64 request";

}

getLogger().error("Found " + typeName

+ ": {} | "

+ "parsedQuery: {} | "

+ "request: {} | "

+ "base 64 request: {} | "

+ "response: {}",

Joiner.on(",").join(statusIds),

requestContext.getParsedQuery().serialize(),

requestContext.getRequest(),

base64Request,

earlybirdResponse);

} else {

getLogger().error("Found " + typeName + ": {} for parsedQuery {}",

Joiner.on(",").join(statusIds),

requestContext.getParsedQuery().serialize());

}

}

}