가장 많은 고객이 선택한 클라우드 모니터링 서비스 1위, 와탭

당신의, IT 서비스의 성능을 극대화합니다.

WhaTap

Docs

Java Agent Plugin Guide

Whatap Support

Version 1.0.2

Table of Contents

Java Agent Plugin 가이드 1

1. 에이전트 옵션 2

1.1. hook_trace_helper_end_patterns 2

1.2. hook_trace_helper_start_patterns 2

1.3. hook_trace_helper_patterns 2

1.4. custom_pool_classes 2

1.5. 설정 예 2

2. Plugin 코드 작성 예 4

2.1. 설정한 메소드 시작시간을 프로파일에 기록하는 예 4

2.2. 설정한 메소드 종료시간과 수행 시간을 프로파일에 기록 하는 예 5

3. Plugin API 7

3.1. 데몬 및 배치 9

3.2. HTTP 서비스 9

3.3. HTTP Outbound 10

3.4. 특정된 메소드 11

3.5. 사용자 정의 Pool 11

4. Plugin 적용 사례 12

4.1. ElasticSearch 엔진의 검색 요청 모니터링 12

Java Agent Plugin 가이드

제목 : Java Agent Plugin Guide
작성자 : Whatap Support
이메일 : support@whatap.io
날짜 : 2019-05-16
버전 : 1.0.2

설명 : Java Agent Plugin 적용 방법에 대해 설명합니다.

와탭에서 제공하는 Plugin 을 활용해 프로파일 정보 내에 부가적인 정보를 추가하거나, 메소드 수행 전/후로 원하는 코드를 주입하는 것 과 같은 다양한 일을 할 수 있습니다.

Chapter 1. 에이전트 옵션

에이전트 플러그인은 메소드 시작/종료 부분에 삽입되어 실행됩니다. 플러그인을 적용할 위치(클래스,메소드 명)를 지정하는 옵션은 다음과 같습니다.

1.1. hook_trace_helper_end_patterns

메소드 종료 부분에 프로파일 플러그인을 삽입할 포인트(클래스 및 메소드명)를 지정합니다.
플러그인 코드는 $WHATAP_HOME/plugin/TraceHelperEnd.x 파일에 작성 합니다.

1.2. hook_trace_helper_start_patterns

default: 메소드 시작 부분에 프로파일 플러그인을 삽일할 포인트(클래스 및 메소드명)를 지정합니다.
플러그인 코드는 $WHATAP_HOME/plugin/TraceHelperStart.x 파일에 작성 합니다.

1.3. hook_trace_helper_patterns

메소드 시작/종료 양쪽에 프로파일 플러그인을 삽입할 포인트(클래스 및 메소드명)를 지정합니다.
플러그인 코드는 $WHATAP_HOME/plugin/TraceHelperStart.x, $WHATAP_HOME/plugin/TraceHelperEnd.x 각각 작성 합니다.

1.4. custom_pool_classes

사용자 정의 Pool을 모니터링 하기위해 Pool사용량 정보를 가진 클래스를 지정 합니다.
플러그인 코드는 $WHATAP_HOME/plugin/CustomPool.x 에 작성 합니다.

1.5. 설정 예

여러개의 클래스를 지정하는 경우 , 로 구분합니다.
패키지명의 문자열 일부/전부를 * 로 치환 할 수 있습니다.

패키지와 메소드 Full Name 을 지정하는 경우

```
whatap.bytecode.instrument.PluginTestA.testA, whatap.bytecode.instrument.PluginTestB.testB
```

패키지명과 메소드 일부를 *로 치환 한 경우

```
*PluginTestA.testA, whatap.bytecode.instrument.PluginTestB.*
```

```
*.testA, *PluginTestB.testB
```

전체를 대상으로 하는 경우

```
*.*
```

Custom Pool 은 식별자@패키지명 형태로 지정합니다.

Custom Pool 을 지정하는 경우

```
whatap_plugin_guide@com.ibm.ws.connectionpool.monitor.ConnectionPoolStats ①
```

① Class 명 앞에 whatap_plugin_guide 라는 식별자를 지정했습니다. 식별자와 클래스는 @로 구분합니다.

Chapter 2. Plugin 코드 작성 예

plugin 코드 작성 예 입니다. 대부분의 경우 Object를 반환하므로 명시적 type casting 처리가 필요합니다.

2.1. 설정한 메소드 시작시간을 프로파일에 기록하는 예

에이전트 hook_trace_helper_patterns 옵션에 적용할 메소드를 정의 합니다.

$WHATAP_HOME/whatap.conf

```
hook_trace_helper_patterns=org.apache.catalina.connector.RequestFacade.*
```

TraceHelperStart.x 에 Plugin 코드를 작성 합니다.

$WHATAP_HOME/plugin/TraceHelperStart.x

```
String prefix = $point.class1 + "." + $point.method; ①
$ctx.setAttribute(prefix + "st", new Long(System.currentTimeMillis())); ②
$ctx.profile(prefix + " Start", new java.util.Date().toString()); ③
```

① String prefix 에 클래스명, 메소드명을 대입합니다.

② 와탭 프로파일 속성명 prefix + "st" 로 현재 시간을 설정 합니다.

③ 프로파일 정보에 현재 시간을 추가 합니다.

프로파일 내역에 메소드 시작 시간이 표시 됩니다.

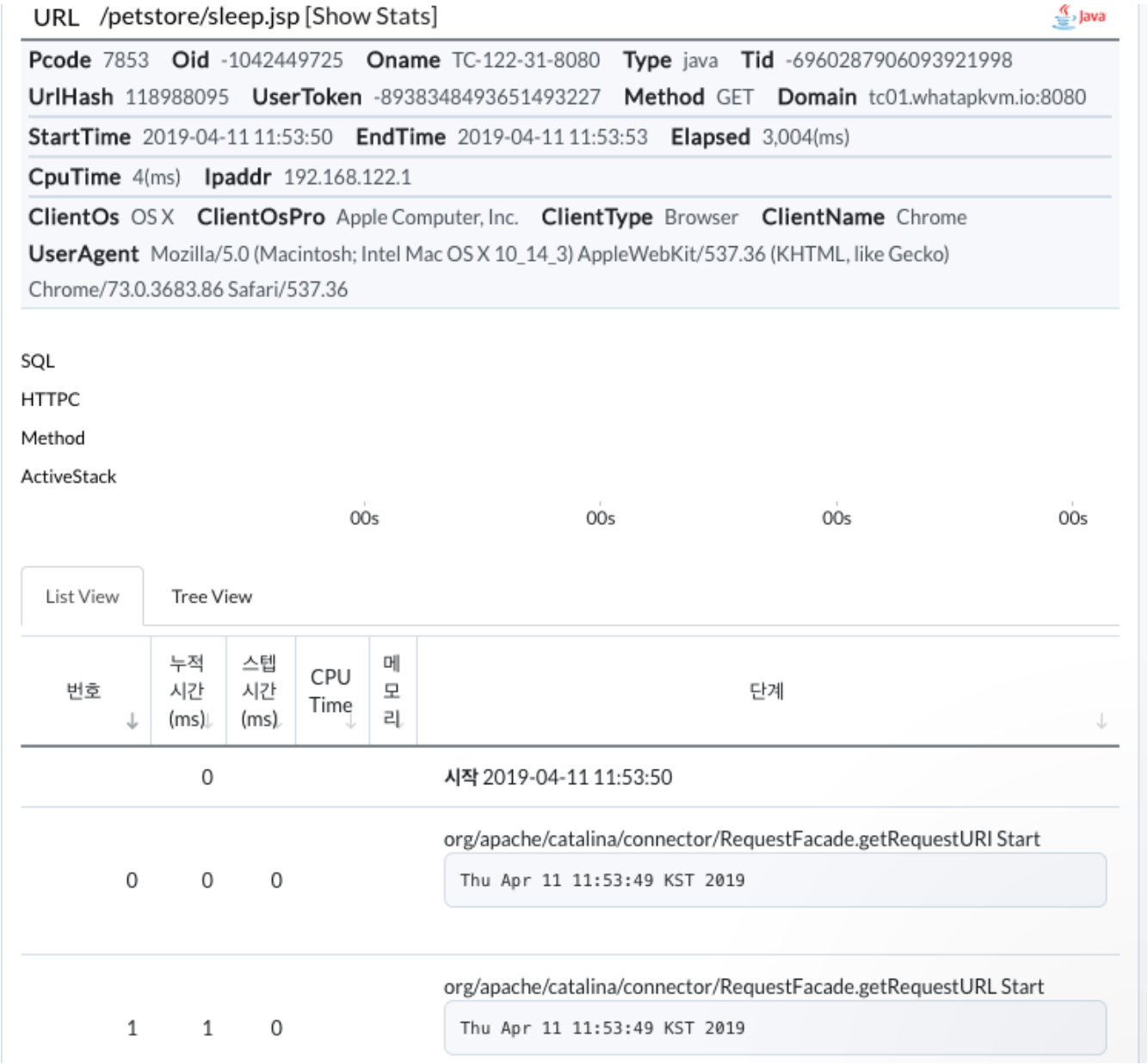

Figure 1. 시작시간 기록된 프로파일 내역

2.2. 설정한 메소드 종료시간과 수행 시간을 프로파일에 기록 하는 예

에이전트 hook_trace_helper_patterns 옵션에 적용할 메소드를 정의 합니다.

$WHATAP_HOME/whatap.conf

```
hook_trace_helper_patterns=org.apache.catalina.connector.RequestFacade.*
```

TraceHelperEnd.x 에 Plugin 코드를 작성 합니다.

$WHATAP_HOME/plugin/TraceHelperEnd.x

```
String prefix = $point.class1 + "." + $point.method;
long st = ((Long) $ctx.getAttribute(prefix + "st")).longValue(); ①
long gap = System.currentTimeMillis() - st;
StringBuilder sb = new StringBuilder();
sb.append(new java.util.Date().toString() + " (Gap:" + gap + " milliseconds)");
$ctx.profile(prefix + " End", sb.toString()); ②
```

① TraceHelperStart.x 에서 추가한 프로파일 속성을 가져 옵니다.

② 프로파일 정보에 시작 시간과의 현재 시간과의 Gap(수행시간)을 추가 합니다.

프로파일 내역에 메소드 종료 시간과 수행 시간이 표시 됩니다.

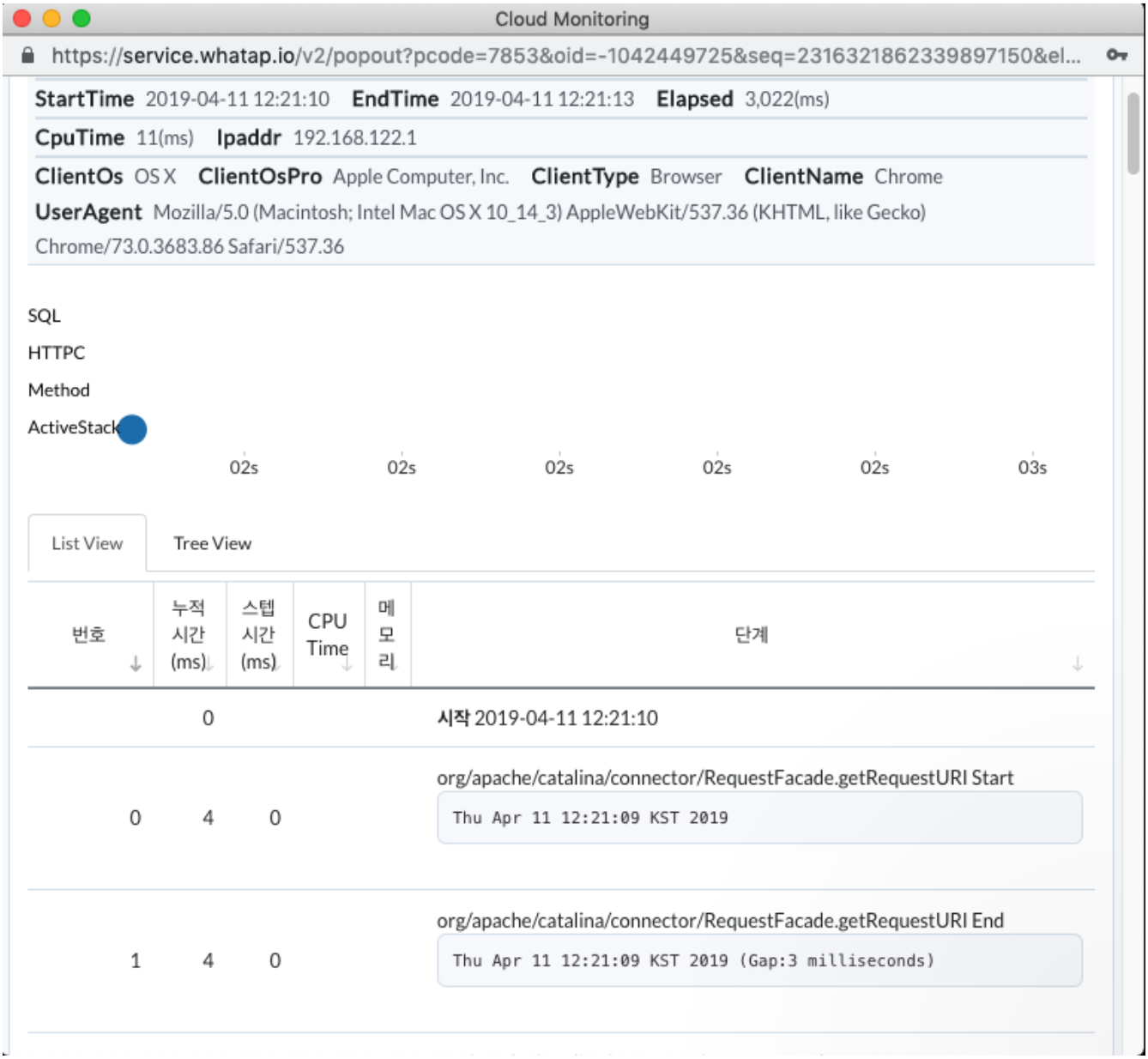

Figure 2. 종료 / 수행시간 까지 기록된 프로파일 내역

Chapter 3. Plugin API

Plugin API 는 공통 항목인 $ctx, $point 와 $pack, $req, $res 로 구분 됩니다.
트랜잭션 시작/종료시, HTTPC 구간, 특정 메소드 실행구간 등에 적용 가능 합니다.

Table 1. 공통 속성

플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
공통				

		$point	String class1	클래스명
			String method	메소드명
			Object this1	Hooking 대상 클래스/메소드
			Object[] args	인자
플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
			Object return	리턴
			void log(Object c)	Logger 를 통한 Logging
			void println(Object c)	System.out.println() 을 통한 출력
			Object field(Object o, String field)	Field 값 반환
			Object method(Object o, String method)	Invoke 메소드
			Object method(Object o, String method, String param)	Invoke 메소드
			String toString(Object o)	Object 를 toSting() 반환
			String toString(Object o, String def)	Object 를 toSting() 반환, null 인경우 def 반환
			int syshash(Object o)	hash 값 반환
			int syshash(HookArgs hook, int x)	x 번째 argument의 hash 값 반환
			int syshash(HookArgs hook)	argument의 hash 값 반환
			int cint(Object o)	int 로 반환
			float cfloat(Object o)	float 으로 반환
			String cString(Object o)	String 으로 반환
			long clong(Object o)	long 으로 반환
			double cdouble(Object o)	double 로 반환
			String desc(Object o)	Class signature 반환
			String toJson(Object o)	json 으로 반환
			void shell(final String cmd, final String env)	shell 실행

3.1. 데몬 및 배치

데몬, 배치와 같은 어플리케이션을 모니터링 할 때 시작점으로 설정한 메소드에 적용하는 API 입니다.

서비스 시작 부분에 적용하는 경우 $WHATAP_HOME/plugin/AppServiceStart.x, 서비스 종료 부분이라면 $WHATAP_HOME/plugin/AppServiceEnd.x 라는 파일 명으로 코드를 작성 합니다.

Table 2. 일반 서비스(데몬 및 배치)

플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
AppServiceStart.x	AppServiceStart.process	$ctx	공통 속성 참조	
		$point	공통 속성 참조	
AppServiceEnd.x	AppServiceEnd.process	$ctx	공통 속성 참조	

3.2. HTTP 서비스

javax.servlet.http.HTTPServlet 을 사용하는 일반적인 웹 어플리케이션에 적용할 수 있는 API 입니다.

서비스 시작 부분에 적용하는 경우 $WHATAP_HOME/plugin/HttpServiceStart.x, 서비스 종료 부분이라면 $WHATAP_HOME/plugin/HttpServiceEnd.x 라는 파일 명으로 코드를 작성 합니다.

Table 3. HTTP 서비스

플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
HttpService Start.x/ HttpService End.x	HttpService Start.process/ HttpService End.process	$ctx	공통 속성 참조	
		$req	String getCookie(String key)	Cookie 값 반환
			String getRequestURI()	RequestURI 반환
			String getRemoteAddr()	RemoteAddr 반환
			String getMethod()	HTTP Method 반환
			String getQueryString()	HTTP QueryString 반환
			String getParameter(String key)	Parameter 값 반환
			Object getAttribute(String key)	Attribute 값 반환
			String getHeader(String key)	Header 값 반환
			Enumeration getParameterNames()	getParameterNames 반환
			Enumeration getHeaderNames()	getHeaderNames 반환
			WrSession getSession()	Session Wrapper 반환
			Set getSessionNames()	getSessionNames 반환
			Object getSessionAttribute(String key)	getSessionAttribute 반환
			boolean isOk()	Plugin 상태 반환
			Throwable error()	Error 반환
		$res	String getContentType()	ContentType 반환
			String getCharacterEncoding()	CharacterEncoding 반환
			boolean isOk()	Plugin 상태 반환
			Throwable error()	Error 반환

3.3. HTTP Outbound

HttpClient와 같은 라이브러를 사용하여 HTTP Outbound Call 수행 할 때 적용할 수 있는 API 입니다.

서비스 시작 부분에 적용하는 경우 $WHATAP_HOME/plugin/HttpCallStart.x, 서비스 종료 부분이라면 $WHATAP_HOME/plugin/HttpCallEnd.x 라는 파일 명으로 코드를 작성 합니다.

Table 4. HTTP Outbound

플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
HttpCallStart.x	HttpCallStart.process	$ctx	공통 속성 참조	
		$req	String url()	URL 반환
			String host()	Hostname 반환
			int port()	Port 반환
			boolean isOk()	Plugin 상태 반환
			Throwable error()	Error 반환

플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
HttpCallEnd.x	HttpCallEnd.process	$ctx	공통 속성 참조	
		$res	String getContentType()	ContentType 반환
			String getCharacterEncoding()	CharacterEncoding 반환
			boolean isOk()	Plugin 상태 반환
			Throwable error()	Error 반환

3.4. 특정된 메소드

와탭 에이전트 옵션 (hook_trace_helper_*) 을 통해 지정한 메소드에 대해 적용하는 API 입니다.

Table 5. 지정한 메소드 구간

플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
TraceHelperStart.x	TraceHelperStart.process	$ctx	공통 속성 참조	
		$point	공통 속성 참조	
TraceHelperEnd.x	TraceHelperEnd.process	$ctx	공통 속성 참조	
		$point	공통 속성 참조	

3.5. 사용자 정의 Pool

사용자 정의 Pool을 모니터링하기 위한 API 입니다.
Pool 사용량 정보를 가진 메소드를 지정하면 와탭에서 모니터링하고 통계정보를 확인 할 수 있습니다. 사례 와 같이 사용됩니다.

Table 6. Custom Pool

플러그인 파일	와탭 패키지	파라미터	메소드/변수	설명
CustomPool.x	CustomPool.process	$id	custom_pool_classes 에 설정한 id 값	
		$pool	custom_pool_classes 에 설정한 class	
		$result	int active(Object o)	Active Pool Count 지정
			int idle(Object o)	Idle Pool Count 지정

Chapter 4. Plugin 적용 사례

실제 Plugin 적용 사례 입니다.

4.1. ElasticSearch 엔진의 검색 요청 모니터링

ElasticSearch 엔진(이하 ES)는 서블릿 엔진이 아니기 때문에 비정형 모니터링을 해야 합니다.
타 제품이 지표 중심의 모니터링을 한다면 와탭의 Plugin 을 활용하면 ES의 요청, 처리시간 그리고 검색 키워드를 프로파일링 할 수 있습니다.

whatap.conf

```
hook_service_patterns=org.elasticsearch.search.SearchService.executeQueryPhase ①
hook_trace_helper_start_patterns=org.elasticsearch.search.query.QueryPhase.execute ②
```

① 트랜잭션 EndPoint로 org.elasticsearch.search.SearchService.executeQueryPhase 를 설정 합니다.

② hook_trace_helper_start_patterns 에 등록된 메소드가 실행 될 때마다 TraceHelperStart.x 내의 코드가 실행 됩니다.

${WHATAP_HOME}/plugin/TraceHelperStart.x

```
if ($ctx.inner()==null){
    return;
}

try {

    String tclass = "org.elasticsearch.search.query.QueryPhase";
    String tmethod = "execute";

    if (tclass.equals($point.class1) && tmethod.equals($point.method)) {
        // 첫번째 argument 의 query 메소드 호출 결과를 String query 에 저장
        String query = " " + method($point.args[0],"query"); ①
        // String query를 프로파일 정보로 출력
        $ctx.profile(query);
    }
} catch(Exception e) {
    $ctx.profile(e.toString());
}
```

① org.elasticsearch.search.query.QueryPhase.execute(SearchContext searchContext) 입니다.
즉, SearchContext.query() 메소드를 invoke 한 결과를 String query 에 저장합니다.