一篇文章让你了解 chaosblade

https://github.com/chaosblade-io/chaosblade-exec-jvm/tree/v0.1.0

验证版本： 0.1.0

以 chaosblade 命令方式进行验证

准备资源

https://github.com/chaosblade-io/chaosblade/releases 下载解压

```
# 启动案例项目
git clone https://github.com/niaoshuai/chaosblade-samples.git
# 找到jvm进程
jps -v
```

基本命令 (blade)

```
$ ./blade
An easy to use and powerful chaos engineering experiment toolkit

Usage:
  blade [command]

Available Commands:
  create      Create a chaos engineering experiment (创建一个实验)
  destroy     Destroy a chaos experiment (销毁一个实验)
  help        Help about any command (各个子命令详细文档)
  prepare     Prepare to experiment (预准备一个实验)
  query       Query the parameter values required for chaos experiments (查询一个
实验)
  revoke      Undo chaos engineering experiment preparation
  status      Query preparation stage or experiment status  (实验状态)
  version     Print version info  (版本)

Flags:
  -d, --debug   Set client to DEBUG mode (设置客户端DEBUG模式)
  -h, --help    help for blade  (帮助)

Use "blade [command] --help" for more information about a command.
```

子命令 (blade prepare/revoke)

```
# 准备为JVM挂载JVM沙箱  24688 自己的JVM进程号 (会有点慢 别看错JVM进程号)
$ ./blade prepare jvm --process 20336
## 结果200 成功 result UID
{"code":200,"success":true,"result":"581b426cda53563e"}
# 取消挂载JVM沙箱 revoke 后面是 UID
$ ./blade revoke 581b426cda53563e
```

子命令 (blade create)

可以查看所有受支持的实验

```
$ ./blade help create
Create a chaos engineering experiment

Usage:
  blade create [command]

Aliases: (命令别名)
  create, c

Examples: (案例)
create dubbo delay --time 3000 --offset 100 --service com.example.Service --
consumer

Available Commands:
  cpu         Cpu experiment  (支持系统CPU实验)
  disk        Disk experiment  (支持系统磁盘实验)
  docker      Execute a docker experiment (支持docker容器实验)
  druid       Druid experiment (支持连接池Druid实验)
  dubbo       dubbo experiment  (支持Dubbo实验)
  http        http experiment (支持Http实验)
  jvm         method (支持JVM实验)
  k8s         Kubernetes experiment (支持K8S实验)
  mysql       mysql experiment (支持MYSQL实验)
  network     Network experiment (支持网络实验)
  process     Process experiment (支持进程实验)
  script      Script chaos experiment   (支持脚本实验脚本)
  servlet     java servlet experiment (支持JAVA Servlet实验)

Flags:
  -h, --help   help for create

Global Flags:
  -d, --debug   Set client to DEBUG mode

Use "blade create [command] --help" for more information about a command.
```

实验 (JVM)

1. OOM

```
# 先查看 JVM 详细支持命令
$ ./blade create jvm -h
# 文章有限 下面我简化
  cpufullload            Process occupied cpu full load (JVM CPU满载)
  delay                  delay time (delay 方式延时)
  outofmemoryerror       JVM out of memory  (OOM)
  return                 Return the specify value  (return)
  script                 Dynamically execute custom scripts  (script)
  throwCustomException   throw custom exception  (抛出自定义异常)
  throwDeclaredException Throw the first declared exception of method  (抛出第一个
声明的异常)

## 查看 JVM  内存OOM 支持
$ ./blade create jvm outofmemoryerror -h
## Flag参数
      --area string             Jvm memory area you want  to cause
OutOfMemoryError,the options:[HEAP, NOHEAP, OFFHEAP] (required) (必填 三个选项)
      --enableSystemGc string   Invoke System.gc() after stop injection,option
value:[true,false],default value is true
  -h, --help                    help for outofmemoryerror
      --process string          Application process name
      --threads string          Thread count to make oom error,if you want to
speed up the oom.default value is :1
      --timeout string          set timeout for experiment

# 注入堆区满载 (请求时间比较短 > 3秒)
 ./blade create jvm outofmemoryerror --area=HEAP
# 注入非堆区满载
 ./blade create jvm outofmemoryerror --area=NOHEAP
# 注入ByteBuffer(OFFHEAP)满载
./blade create jvm outofmemoryerror --area=OFFHEAP

## 每次执行实验成功都会返回UID
{"code":200,"success":true,"result":"786e761471352e01"}
## 销毁也会有相应的提示
{"code":200,"success":true,"result":"command: jvm outofmemoryerror --help false
--area HEAP --debug false"}
```

2. JVM CPU 满载

```
# 执行命令 c 是create 的简化命令
$ ./blade c jvm cpufullload
# 结果
{"code":200,"success":true,"result":"e7ce7cca9efd7c65"}
## 通过命令 或者系统任务管理器可一查看

# 取消/销毁实验 destroy 后面是 UID
./blade destroy e7ce7cca9efd7c65
# 结果
{"code":200,"success":true,"result":"command: jvm cpufullload --debug false --
help false"}
## 通过命令 或者系统任务管理器可一查看
```

实验 (Servlet)

自定义异常

```
# 抛出自定义异常
/blade create servlet throwCustomException --exception=java.lang.Exception --
servletpath=/index --pathinfo=/rest --method=get
## 日志省略
```

延时

```
# 延时 servletpath 就是 servlet的路径   pathinfo 可以理解为springmvc里面路径参数
(PathVariable)
./blade create servlet delay --time=2000 --servletpath=/index --pathinfo=/rest -
-method=get
{"code":200,"success":true,"result":"fbc3b18e76d61dd9"}
# 测试 并销毁
./blade destroy fbc3b18e76d61dd9
```

实验 （Http）

](https://github.com/chaosblade-io/chaosblade-exec-jvm/wiki/开发者指南-协议篇)

```
# 状态
http://localhost:3658/sandbox/default/module/http/chaosblade/status?
suid=su378dsn137s53bs8adcn
# 创建
## http 之 rest 调用 (org.springframework.web.client.RestTemplate)
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=http&action=delay&time=3000&uri=http://www.bai
du.com&rest=true
## dubbo 调用
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
uid=su378dsn137s53bs8adcn&target=dubbo&action=delay&time=3000&service=com.exampl
e.HelloService&version=1.0.0&consumer=true

# 销毁
http://localhost:3658/sandbox/default/module/http/chaosblade/destroy?
suid=su378dsn137s53bs8adcn
```

1. HTTP REST

```
// 准备一个REST调用
public static void getBaidu(){
    RestTemplate restTemplate = new RestTemplate();
    restTemplate.getForObject("http://www.baidu.com", String.class);
}
```

```
# 开启一个Http Rest 混沌实验
$ ./blade create http delay --time=3000  --rest --uri=http://www.baidu.com
# 验证结果 OK 使用完成之后记得销毁
```

2. HTTP CLIENT3

```
// 准备一个httpClient3调用
public static void getBaidu() throws IOException {
     HttpClient client = new HttpClient();
     GetMethod get = new GetMethod("http://www.baidu.com");
     // execute method and handle any error responses.
     client.executeMethod(get);
     get.getResponseBodyAsString();
     get.releaseConnection();
}
```

```
# 开启一个HTTP CLIENT3 混沌 [延时]
# [备注 当前版本需要 uri在没有参数的情况要加上最后面的 / 后面版本会修复这个问题 ]
./blade create http delay --time=3000  --httpclient3 --uri=http://www.baidu.com/
# 验证结果 OK 使用完成之后记得销毁
```

```
# 开启一个HTTP CLIENT3 混沌实验 [自定义异常]
# [备注 当前版本需要 uri在没有参数的情况要加上最后面的 / 后面版本会修复这个问题 ]
./blade create http throwCustomException --exception=java.lang.Exception  --
httpclient3 --uri=http://www.baidu.com/
# 验证结果 OK 使用完成之后记得销毁
```

3. HTTP CLIENT4

```
//准备一个httpclient4调用
public static void getBaidu() throws IOException {
    HttpClient client = HttpClients.createDefault();
    HttpGet get = new HttpGet();
    get.setURI(URI.create("http://www.baidu.com"));
    HttpResponse response = client.execute(get);
    StatusLine statusLine = response.getStatusLine();
    System.out.println(statusLine);
}
```

```
# 开启一个HTTP CLIENT4 混沌实验
./blade create http delay --time=3000  --httpclient4 --uri=http://www.baidu.com
# 验证结果 OK 使用完成之后记得销毁
```

以JVM Sandbox 方式进行验证

找到 jvm.spec.yaml 看查看支持的混沌注入参数

准备资源

```
# 准备示例项目 并自己手动启动
git clone https://github.com/niaoshuai/chaosblade-samples.git
## 启动项目的时候 可增加启动参数（用于远程Debug chaosblade-exec-jvm 源码）
-agentlib:jdwp=transport=dt_socket,address=127.0.0.1:5005,suspend=n,server=y

# 移动chaosblade-java-agent-0.1.0.jar 到 ~/.sandbox-module/
```

```
cp chaosblade-java-agent-0.1.0.jar ~/.sandbox-module/
# 启动并查看chaosblade 是否成功被挂载
bash ~/sandbox/bin/sandbox.sh -p 4314 -P 3658 -l
# 日志如下  FROZEN 默认是被冻结
chaosblade  FROZEN      LOADED      0       0       0.1.0 Changjun Xiao
# 找到目标应用的进程号
jps -v
# 使用沙箱attach p参数是进程号  P参数是沙箱对外通信端口 a参数是激活模块
bash ~/sandbox/bin/sandbox.sh -p 4314 -P 3658 -a chaosblade
```

内置URL

```
# 创建实验 (额外参数参考下面案例)
http://localhost:3658/sandbox/default/module/http/chaosblade/create?suid=
# 实验状态
http://localhost:3658/sandbox/default/module/http/chaosblade/status?suid=
# 销毁实验
http://localhost:3658/sandbox/default/module/http/chaosblade/destroy?suid=
```

实验 （JVM）

1. OOM

```
# 注入堆区满载 (请求时间比较短 > 3秒)
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=jvm&action=OutOfMemoryError&area=HEAP&threads=
1&enableSystemGc=false
# 注入非堆区满载
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=jvm&action=OutOfMemoryError&area=NOHEAP&thread
s=1&enableSystemGc=false
# 注入ByteBuffer(OFFHEAP)满载
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=jvm&action=OutOfMemoryError&area=OFFHEAP&threa
ds=1&enableSystemGc=false
```

2. JVM CPU 满载

```
# 注入JVM CPU满载
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=jvm&action=cpufullload
```

实验 (Servlet)

自定义异常

```
# 抛出自定义异常
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=servlet&action=throwCustomException&exception=
java.lang.Exception&servletpath=/index&pathinfo=/rest&method=get
```

延时

```
# 延时
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=servlet&action=delay&time=1000&servletpath=/in
dex&pathinfo=/rest&method=get
```

实验 （Http）

分析源码

https://github.com/chaosblade-io/chaosblade-exec-jvm/wiki/%E5%BC%80%E5%8F%91%E8%80%85%E6%8C%87%E5%8D%97-%E5%8D%8F%E8%AE%AE%E7%AF%87

```
# 状态
http://localhost:3658/sandbox/default/module/http/chaosblade/status?
suid=su378dsn137s53bs8adcn
# 创建
## http 之 rest 调用 (org.springframework.web.client.RestTemplate)
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=http&action=delay&time=3000&uri=http://www.bai
du.com&rest=true
## dubbo 调用
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
uid=su378dsn137s53bs8adcn&target=dubbo&action=delay&time=3000&service=com.exampl
e.HelloService&version=1.0.0&consumer=true

# 销毁
http://localhost:3658/sandbox/default/module/http/chaosblade/destroy?
suid=su378dsn137s53bs8adcn
```

1. HTTP REST

```
// 准备一个REST调用
public static void getBaidu(){
    RestTemplate restTemplate = new RestTemplate();
    restTemplate.getForObject("http://www.baidu.com", String.class);
}
```

```
# 开启一个Http Rest 混沌实验
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=http&action=delay&time=3000&uri=http://www.bai
du.com&rest=true
# 验证结果 OK 使用完成之后记得销毁
```

2. HTTP CLIENT3

```
// 准备一个httpClient3调用
public static void getBaidu() throws IOException {
     HttpClient client = new HttpClient();
     GetMethod get = new GetMethod("http://www.baidu.com");
     // execute method and handle any error responses.
     client.executeMethod(get);
     get.getResponseBodyAsString();
     get.releaseConnection();
 }
```

```
# 开启一个HTTP CLIENT3 混沌实验 [延时]
# [备注 当前版本需要 uri在没有参数的情况要加上最后面的 / 后面版本会修复这个问题 ]
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=http&action=delay&time=3000&uri=http://www.bai
du.com/&httpclient3=true
# 验证结果 OK 使用完成之后记得销毁
```

```
# 开启一个HTTP CLIENT3 混沌实验 [自定义异常]
# [备注 当前版本需要 uri在没有参数的情况要加上最后面的 / 后面版本会修复这个问题 ]
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=http&&action=throwCustomException&exception=ja
va.lang.Exception&uri=http://www.baidu.com/&method=get&httpclient3=true
# 验证结果 OK 使用完成之后记得销毁
```

3. HTTP CLIENT4

```
//准备一个httpclient4调用
public static void getBaidu() throws IOException {
    HttpClient client = HttpClients.createDefault();
    HttpGet get = new HttpGet();
    get.setURI(URI.create("http://www.baidu.com"));
    HttpResponse response = client.execute(get);
    StatusLine statusLine = response.getStatusLine();
    System.out.println(statusLine);
}
```

```
# 开启一个HTTP CLIENT4 混沌实验
http://localhost:3658/sandbox/default/module/http/chaosblade/create?
suid=su378dsn137s53bs8adcn&target=http&action=delay&time=3000&uri=http://www.bai
du.com&httpclient4=true
# 验证结果 OK 使用完成之后记得销毁
```