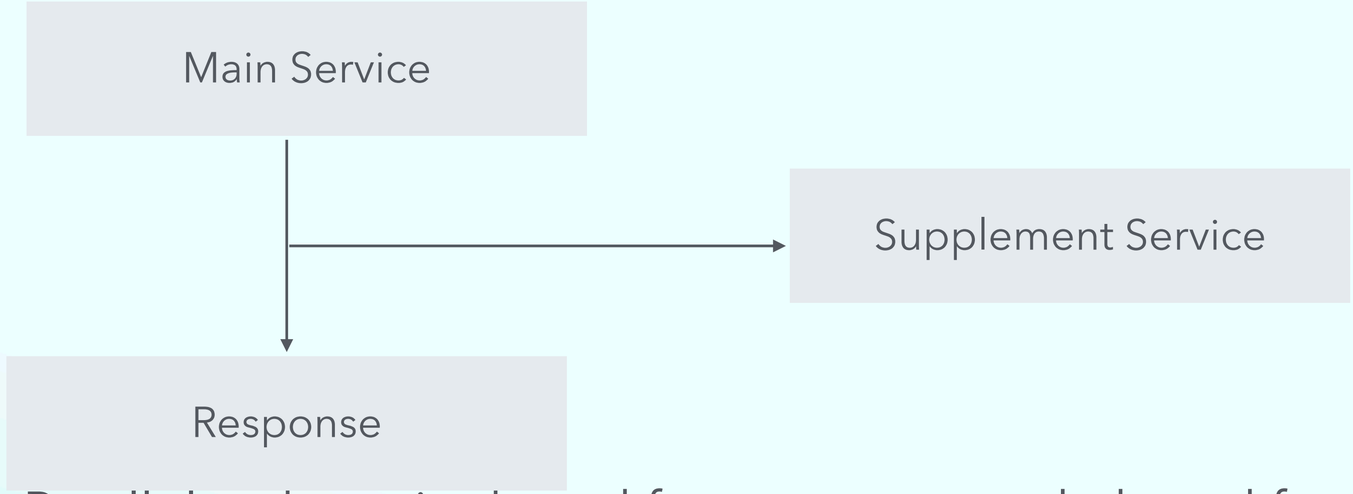
ThreadPool Research

Agenda

- Thread Implemented by Java / How to use it in service/Why ThreadPool
- Configuration of ThreadPool /How to Implemented the ThreadPool in service
- Q&A

Thread Implemented by Java

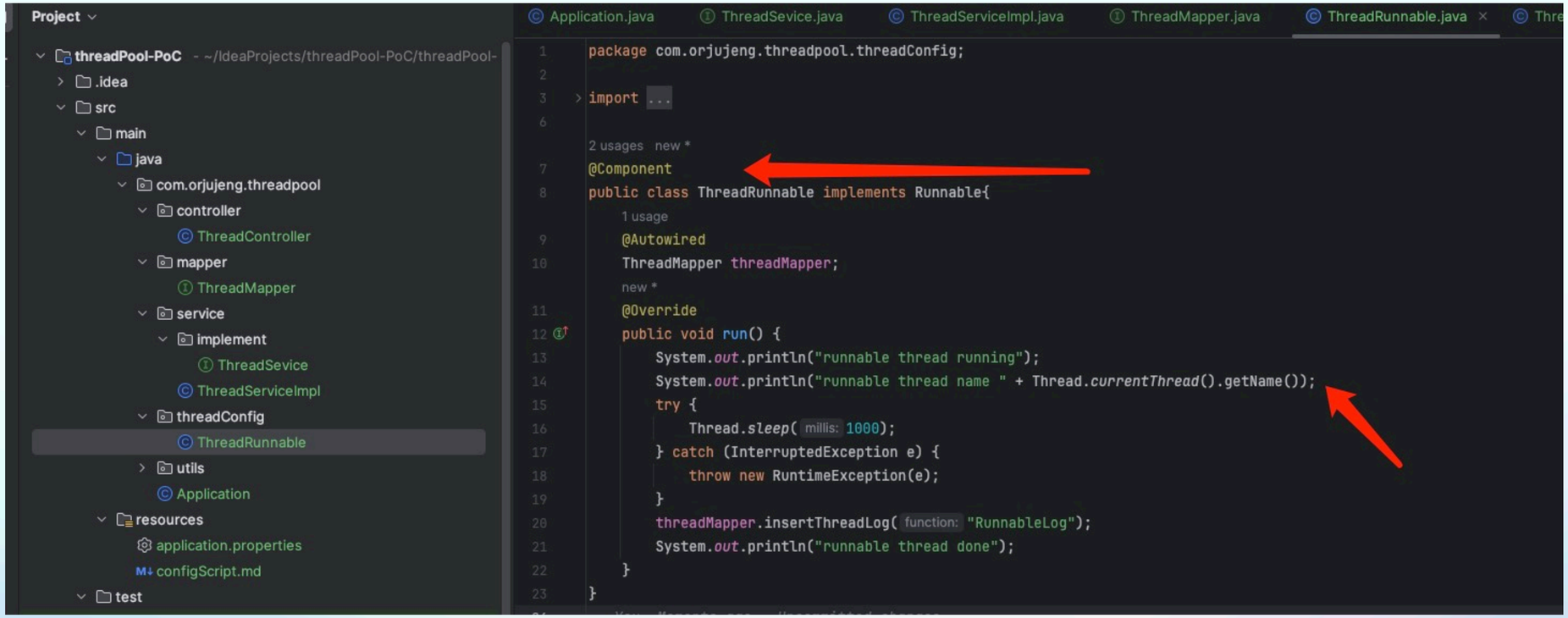
1. Main thread finish the basic logic and response to frontend ASAP



2. Parallel task: main thread for summary work thread for finish the request respective

Thread Implemented by Java

- Thread runnable/ static proxy

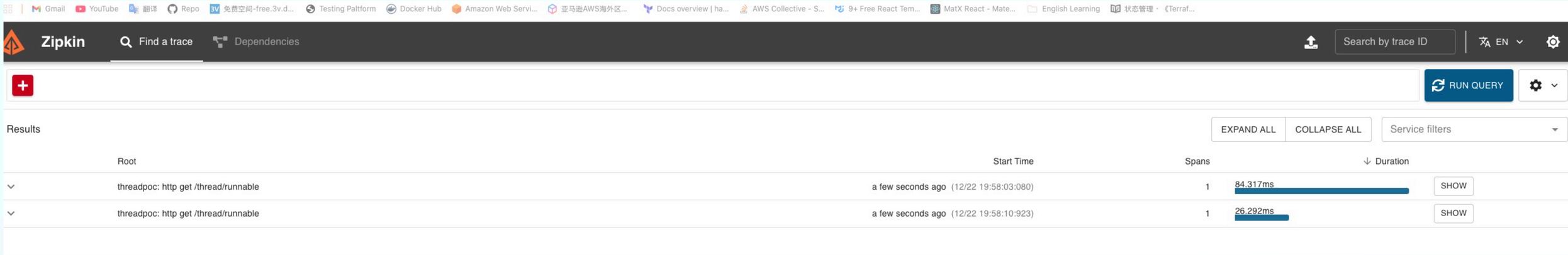


Thread Implemented by Java

```java
    @Override
    public Response runnableService() {
        System.out.println("Main thread executing");
        Thread insertLog = new Thread(threadRunnable);
        insertLog.start();
        try {
            insertLog.join();   You, 27 minutes ago • Uncommitted changes
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        // insert log
        System.out.println("Main thread Done");
        return Response.success( data: "service response data");
    }
}
```

Thread Implemented by Java

1. Main thread finish the basic logic and response to frontend ASAP




```
2024-12-22T19:58:03.071+08:00  INFO 8803 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 7 ms
Main thread executing
Main thread Done
runnable thread running
runnable thread name Thread-7
2024-12-22T19:58:04.192+08:00  INFO 8803 --- [       Thread-7] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-22T19:58:04.519+08:00  INFO 8803 --- [       Thread-7] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@47b2eebf
2024-12-22T19:58:04.525+08:00  INFO 8803 --- [       Thread-7] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
runnable thread done
Main thread executing
Main thread Done
runnable thread running
runnable thread name Thread-8
runnable thread done
```

Thread Implemented by Java

2. Parallel task: main thread for summary work thread for finish the request respective

```java
public Response runnableService() {
    System.out.println("Main thread executing");
    Thread insertLog = new Thread(threadRunnable);
    insertLog.start();
    try {
        insertLog.join();   You, Moments ago • Uncommitted changes
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    // insert log
    System.out.println("Main thread Done");
    return Response.success( data: "service response data");
}
}
```

```
2024-12-22T20:02:45.043+08:00  INFO 8858 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 8 ms
Main thread executing
runnable thread running
runnable thread name Thread-7
2024-12-22T20:02:46.159+08:00  INFO 8858 --- [       Thread-7] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-22T20:02:46.484+08:00  INFO 8858 --- [       Thread-7] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@2b8a9fc
2024-12-22T20:02:46.488+08:00  INFO 8858 --- [       Thread-7] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
runnable thread done
Main thread Done
Main thread executing
runnable thread running
runnable thread name Thread-8
runnable thread done
Main thread Done
```

Zipkin | Find a trace | Dependencies | Search by trace ID | EN

RUN QUERY

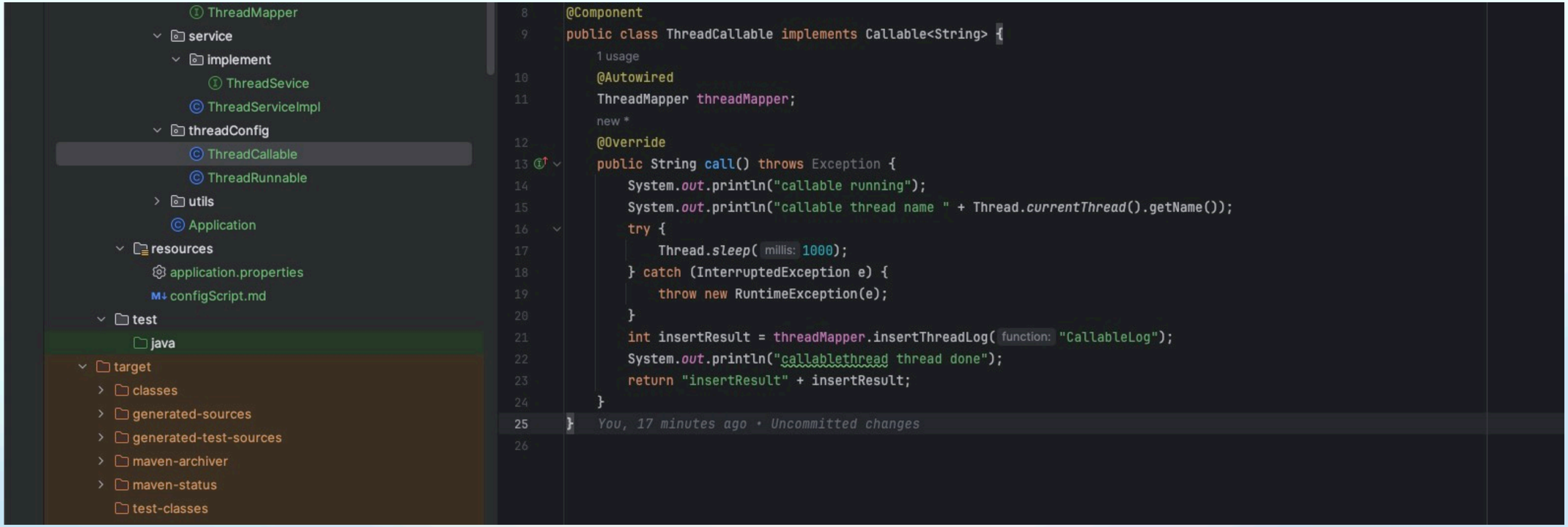
Results

EXPAND ALL | COLLAPSE ALL | Service filters

Root	Start Time	Spans	Duration	
threadpoc: http get /thread/runnable	a few seconds ago (12/22 20:02:45:055)	1	1.532s	SHOW
threadpoc: http get /thread/runnable	a few seconds ago (12/22 20:02:47:856)	1	1.053s	SHOW

Thread Implemented by Java

- Thread callable



Thread Implemented by Java

- Thread callable

1. Main thread finish the basic logic and response to frontend ASAP

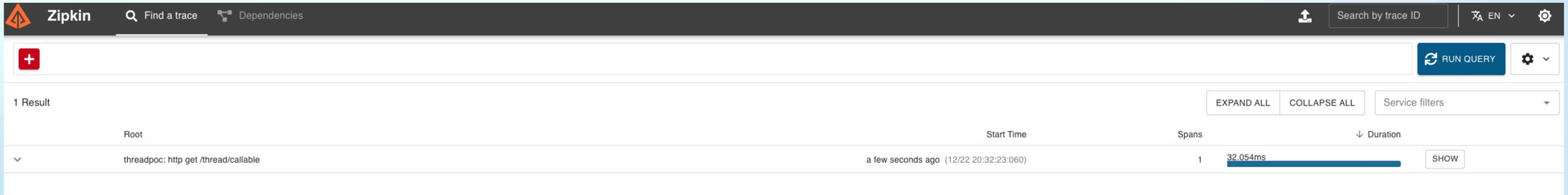
```java
1 usage  new *
@Override
public Response callableService() {
    System.out.println("Main thread executing");
    FutureTask<String> ft = new FutureTask<>(threadCallable);
    Thread insertLog = new Thread(ft);
    // insert log
    insertLog.start();
    System.out.println("Main thread Done");   You, 10 minutes ago • Uncommitted changes
    return Response.success( data: "service response data");
}
}
```

```
Main thread Done
callable running
callable thread name Thread-8
callablethread thread done
Main thread executing
Main thread Done
callable running
callable thread name Thread-9
callablethread thread done
Main thread executing
Main thread Done
callable running
callable thread name Thread-10
callablethread thread done
```

Thread Implemented by Java

- Thread callable

```java
1 usage  new *
@Override
public Response callableService() {
    System.out.println("Main thread executing");
    FutureTask<String> ft = new FutureTask<>(threadCallable);
    Thread insertLog = new Thread(ft);
    // insert log
    insertLog.start();
    System.out.println("Main thread Done");   You, 10 minutes ago • Uncommitted changes
    return Response.success( data: "service response data");
}
}
```



Thread Implemented by Java

2. Parallel task: main thread for summary work thread for finish the request respective

```java
1 usage  new *
@Override
public Response callableService() {
    System.out.println("Main thread executing");
    FutureTask<String> ft = new FutureTask<>(threadCallable);
    Thread insertLog = new Thread(ft);
    // insert log
    insertLog.start();
    try {
        System.out.println(ft.get());
    } catch (Exception e) {
        e.printStackTrace();   You, Moments ago • Uncommitted changes
    }
    System.out.println("Main thread Done");
    return Response.success( data: "service response data");
}
}
```

```
Main thread executing
callable running
callable thread name Thread-7
2024-12-22T20:36:59.510+08:00  INFO 9288 --- [       Thread-7] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-22T20:36:59.840+08:00  INFO 9288 --- [       Thread-7] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@5e5ae23f
2024-12-22T20:36:59.845+08:00  INFO 9288 --- [       Thread-7] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
callablethread thread done
insertResult1
Main thread Done
Main thread executing
callable running
callable thread name Thread-8
callablethread thread done
insertResult1
Main thread Done
```

Thread Implemented by Java

2. Parallel task: main thread for summary work thread for finish the request respective

```java
1 usage  new *
@Override
public Response callableService() {
    System.out.println("Main thread executing");
    FutureTask<String> ft = new FutureTask<>(threadCallable);
    Thread insertLog = new Thread(ft);
    // insert log
    insertLog.start();
    try {
        System.out.println(ft.get());
    } catch (Exception e) {
        e.printStackTrace();   You, Moments ago • Uncommitted changes
    }
    System.out.println("Main thread Done");
    return Response.success( data: "service response data");
}
}
```

	id	function	insert_date
1	2,025	CallableLog	2024-12-22 20:37:12.221318
2	2,024	CallableLog	2024-12-22 20:36:59.857968
3	2,023	CallableLog	2024-12-22 20:32:24.100251
4	2,022	CallableLog	2024-12-22 20:31:02.649426
5	2,021	CallableLog	2024-12-22 20:21:48.557869
6	2,020	CallableLog	2024-12-22 20:21:46.422968
7	2,019	RunnableLog	2024-12-22 20:02:48.890791
8	2,018	RunnableLog	2024-12-22 20:02:46.503733
9	2,017	RunnableLog	2024-12-22 19:58:11.967083
10	2,016	RunnableLog	2024-12-22 19:58:04.537006
11	2,015	RunnableLog	2024-12-22 19:15:40.097793
12	2,014	RunnableLog	2024-12-22 19:15:37.601609
13	2,013	RunnableLog	2024-12-22 19:15:34.367396
14	2,012	RunnableLog	2024-12-22 19:13:48.850408
15	2,011	RunnableLog	2024-12-22 19:13:16.490489
16	2,010	RunnableLog	2024-12-22 19:13:15.510087
17	2,009	RunnableLog	2024-12-22 19:13:14.323628
18	2,008	RunnableLog	2024-12-22 19:13:12.073773
19	2,007	RunnableLog	2024-12-22 19:11:11.221506
20	2,006	RunnableLog	2024-12-22 19:11:10.594707
21	2,005	RunnableLog	2024-12-22 19:11:03.199144
22	2,004	RunnableLog	2024-12-22 19:10:23.422011
23	2,003	RunnableLog	2024-12-22 19:10:21.786977
24	2,002	RunnableLog	2024-12-22 19:10:20.253948
25	2,001	RunnableLog	2024-12-22 16:25:45.809715

Root	Start Time	Spans	Duration	
threadpoc: http get /thread/callable	a few seconds ago (12/22 20:36:58:414)	1	1.532s	SHOW
threadpoc: http get /thread/callable	a few seconds ago (12/22 20:37:11:175)	1	1.061s	SHOW
threadpoc: http get /thread/callable	5 minutes ago (12/22 20:32:23:060)	1	32.054ms	SHOW

Thread Implemented by Java

- Thread @Async no any thread or thread pool config

```
1 usage  new *
@Override
@Async
public void asyncService() {
    System.out.println("async thread executing");
    System.out.println("async thread name " + Thread.currentThread().getName());
    threadMapper.insertThreadLog( function: "async");
    System.out.println("async thread Done");
}   You, 2 minutes ago • Uncommitted changes
}
```

```
async Main thread name http-nio-8080-exec-3
async thread executing
async thread name task-2
async thread Done
```

Root	Start Time	Spans	↓ Duration	
threadpoc: http get /thread/async	a few seconds ago (12/22 20:54:01:977)	1	79.060ms	SHOW
threadpoc: http get /thread/async	a few seconds ago (12/22 20:54:03:927)	1	21.494ms	SHOW

Thread Implemented by Java

- Thread @Async no any thread or thread pool config

```java
1 usage  new *
@Override
@Async
public void asyncService()  {
    System.out.println("async thread executing");
    System.out.println("async thread name " + Thread.currentThread().getName());
    try {
        Thread.sleep( millis: 1000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    threadMapper.insertThreadLog( function: "async");
    System.out.println("async thread Done");
}   You, Moments ago • Uncommitted changes
}
```

threadpoc: http get /thread/async	a few seconds ago (12/22 21:00:21:850)	1	99.380ms
threadpoc: http get /thread/async	a few seconds ago (12/22 21:00:24:543)	1	15.990ms
threadpoc: http get /thread/async	a few seconds ago (12/22 21:00:23:288)	1	14.245ms

SimpleAsyncTaskExecutor

```
2024-12-22T21:00:23.338+08:00  INFO 9614 --- [         task-1] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@51cf8ccd
2024-12-22T21:00:23.343+08:00  INFO 9614 --- [         task-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
async thread Done
async thread Done
async Main thread name http-nio-8080-exec-5
async thread executing
async thread name task-3
async thread Done
```

1. Main thread finish the basic logic and response to frontend ASAP

Thread Implemented by Java

- Thread @Async no any thread or thread pool config

```java
@Override
@Async
public CompletableFuture<String> asyncService()  {
    System.out.println("async thread executing");
    System.out.println("async thread name " + Thread.currentThread().getName());
    try {
        Thread.sleep( millis: 1000);
    } catch (InterruptedException e) {
        throw new RuntimeException(e);
    }
    int insertResult = threadMapper.insertThreadLog( function: "async");
    System.out.println("async thread Done");
    return new CompletableFuture<String>().completedFuture( value: "insertResult "+ insertResult);
}
}   You, 2 minutes ago • Uncommitted changes
```

```java
@RequestMapping(value = "/async",method = RequestMethod.GET)
public Response async() throws ExecutionException, InterruptedException {
    System.out.println("async Main thread name " + Thread.currentThread().getName());
    CompletableFuture result = threadSevice.asyncService();
    return Response.success( data: "controller data" + result.get());   You, 3 minutes ago • Uncommitted changes
}
```

Root	Start Time	Spans	Duration	
threadpoc: http get /thread/async	a few seconds ago (12/22 21:24:58:914)	1	1.520s	SHO
threadpoc: http get /thread/async	a few seconds ago (12/22 21:25:08:528)	1	1.056s	SHO

```
2024-12-22T21:25:00.024+08:00  INFO 9893 --- [         task-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-22T21:25:00.334+08:00  INFO 9893 --- [         task-1] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.
2024-12-22T21:25:00.337+08:00  INFO 9893 --- [         task-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
async thread Done
async Main thread name http-nio-8080-exec-2
async thread executing
async thread name task-2
async thread Done
```

2. Parallel task: main thread for summary work thread for finish the request respective

Thread Implemented by Java

- CompletableFuture

	id	function	insert_date
1	2,036	CompletableFuture	2024-12-22 22:02:35.948141
2	2,035	CompletableFuture	2024-12-22 22:00:10.117295
3	2,034	CompletableFuture	2024-12-22 21:58:45.776139

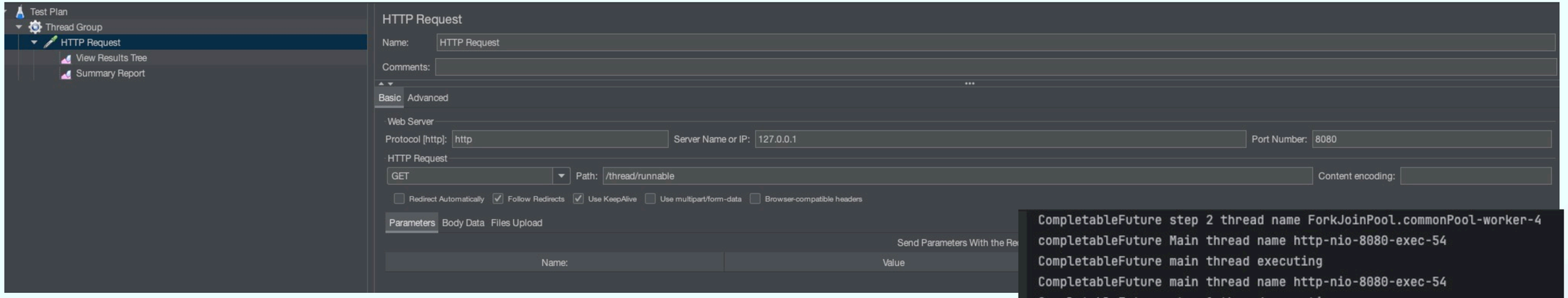
```java
1 usage  new *
@Override
public Response completableFutureService() throws ExecutionException, InterruptedException {
    System.out.println("CompletableFuture main thread executing");
    System.out.println("CompletableFuture main thread name " + Thread.currentThread().getName());
    CompletableFuture<Integer> step1 = CompletableFuture.supplyAsync(() -> {
        System.out.println("CompletableFuture step 1 thread executing");
        System.out.println("CompletableFuture step 1 thread name " + Thread.currentThread().getName());
        try {
            Thread.sleep( millis: 2000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return 1;
    });

    CompletableFuture<Integer> step2 = CompletableFuture.supplyAsync(() -> {
        System.out.println("CompletableFuture step 2 thread executing");
        System.out.println("CompletableFuture step 2 thread name " + Thread.currentThread().getName());
        int insertResult = threadMapper.insertThreadLog( function: "CompletableFuture");
        try {
            Thread.sleep( millis: 2000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return insertResult;
    });
    CompletableFuture<Void> allOf = CompletableFuture.allOf(step1,step2);

    CompletableFuture<Integer> resultFuture = allOf.thenApply(v -> {   You, 2 minutes ago • Uncommitted changes
        Integer result1 = step1.join();
        Integer result2 = step2.join();
        return result1 + result2;
    });

    Integer result = resultFuture.get();
    return Response.success(result);
}
}
```

Thread Implemented by Java

2. Parallel task: main thread for summary work thread for finish the request respective

```
CompletableFuture main thread name http-nio-8080-exec-1
CompletableFuture main thread executing
CompletableFuture main thread name http-nio-8080-exec-1
CompletableFuture step 1 thread executing
CompletableFuture step 2 thread executing
CompletableFuture step 1 thread name ForkJoinPool.commonPool-worker-1
CompletableFuture step 2 thread name ForkJoinPool.commonPool-worker-2
2024-12-22T22:02:35.618+08:00  INFO 10340 --- [onPool-worker-2] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-22T22:02:35.928+08:00  INFO 10340 --- [onPool-worker-2] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@3aea1d7d
2024-12-22T22:02:35.932+08:00  INFO 10340 --- [onPool-worker-2] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
```

Thread Implemented by Java




```
runnable thread name Thread-1003
runnable thread running
runnable thread name Thread-1004
runnable thread running
runnable thread name Thread-1005
runnable thread running
runnable thread name Thread-1006
runnable thread done
runnable thread done
Main thread Done
Main thread Done
runnable thread done
Main thread Done
```

```
CompletableFuture step 2 thread name ForkJoinPool.commonPool-worker-4
completableFuture Main thread name http-nio-8080-exec-54
CompletableFuture main thread executing
CompletableFuture main thread name http-nio-8080-exec-54
CompletableFuture step 1 thread executing
CompletableFuture step 1 thread name ForkJoinPool.commonPool-worker-2
CompletableFuture step 2 thread executing
CompletableFuture step 2 thread name ForkJoinPool.commonPool-worker-6
completableFuture Main thread name http-nio-8080-exec-40
completableFuture Main thread name http-nio-8080-exec-68
CompletableFuture main thread executing
CompletableFuture main thread executing
CompletableFuture main thread name http-nio-8080-exec-40
CompletableFuture main thread name http-nio-8080-exec-68
CompletableFuture step 1 thread executing
CompletableFuture step 1 thread name ForkJoinPool.commonPool-worker-4
CompletableFuture step 2 thread executing
CompletableFuture step 2 thread executing
CompletableFuture step 2 thread name ForkJoinPool.commonPool-worker-2
CompletableFuture step 1 thread executing
CompletableFuture step 1 thread name ForkJoinPool.commonPool-worker-7
CompletableFuture step 2 thread name ForkJoinPool.commonPool-worker-6
completableFuture Main thread name http-nio-8080-exec-41
CompletableFuture main thread executing
CompletableFuture main thread name http-nio-8080-exec-41
CompletableFuture step 1 thread executing
CompletableFuture step 1 thread name ForkJoinPool.commonPool-worker-2
CompletableFuture step 2 thread executing
CompletableFuture step 2 thread name ForkJoinPool.commonPool-worker-6
```

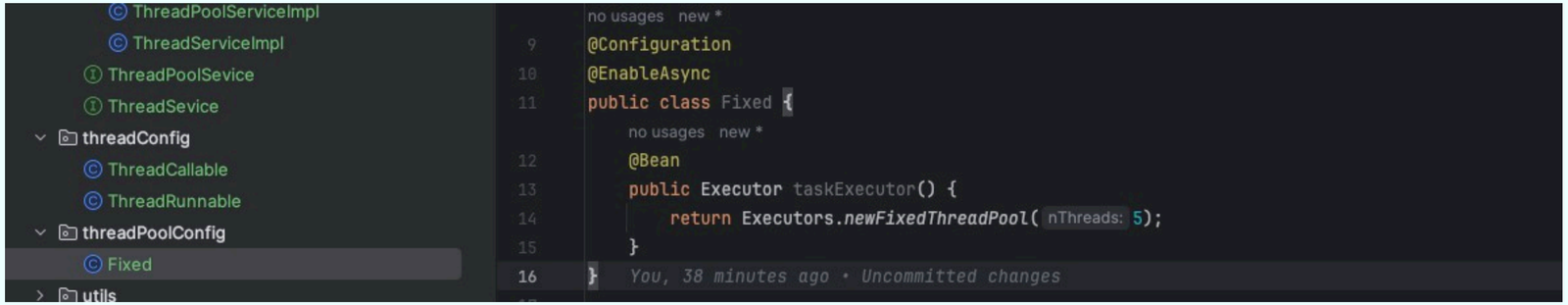
Thread Implemented by Java

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	8570	3868	2	271732	17276.63	1.17%	22.1/min	0.10	0.07	268.9
TOTAL	8570	3868	2	271732	17276.63	1.17%	22.1/min	0.10	0.07	268.9

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	7570	4279	2	271732	18341.24	1.32%	19.6/min	0.09	0.06	271.8
TOTAL	7570	4279	2	271732	18341.24	1.32%	19.6/min	0.09	0.06	271.8

特性	使用线程池	不使用线程池
线程管理	线程池复用线程，减少线程创建和销毁的开销。	每次任务执行时都会创建新的线程，增加了创建销毁的开销。
性能	提高系统性能，减少线程创建和销毁的时间。	高并发时性能较差，线程创建和销毁频繁。
资源管理	控制最大线程数和线程队列，避免系统过载。	无法控制线程数，可能导致资源消耗过多。
调度和执行	提供任务队列，任务按照策略调度执行。	任务的调度完全依赖操作系统，缺乏灵活性。
错误处理	提供统一的错误处理和任务恢复机制。	错误处理复杂，需要手动管理。
灵活性	灵活配置线程池大小，适应不同的任务需求。	缺乏灵活性，无法动态调整线程数。
适用场景	高并发、大规模任务处理、任务调度复杂的场景。	低并发任务或任务量较少的场景。

Configuration of ThreadPool




```java
public Response completableFutureService() throws ExecutionException, InterruptedException {
    System.out.println("CompletableFuture pool main thread executing");
    System.out.println("CompletableFuture pool main thread name " + Thread.currentThread().getName());
    CompletableFuture<Integer> step1 = CompletableFuture.supplyAsync(() -> {
        System.out.println("CompletableFuture pool step 1 thread executing");
        System.out.println("CompletableFuture pool step 1 thread name " + Thread.currentThread().getName());
        try {
            Thread.sleep( millis: 1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return 1;
    },executor);
```

Configuration of ThreadPool

newFixedThreadPool

```
async threadpool executing
async threadpool name pool-3-thread-1
2024-12-22T22:54:19.771+08:00  INFO 10960 --- [pool-3-thread-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-22T22:54:20.103+08:00  INFO 10960 --- [pool-3-thread-1] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@2740dabd
2024-12-22T22:54:20.107+08:00  INFO 10960 --- [pool-3-thread-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
async threadpool Done
async Main thread name http-nio-8080-exec-2
async threadpool executing
async threadpool name pool-3-thread-2
async threadpool Done
async Main thread name http-nio-8080-exec-4
async threadpool executing
async threadpool name pool-3-thread-3
async threadpool Done
async Main thread name http-nio-8080-exec-3
async threadpool executing
async threadpool name pool-3-thread-4
async threadpool Done
async Main thread name http-nio-8080-exec-5
async threadpool executing
async threadpool name pool-3-thread-5
async threadpool Done
async Main thread name http-nio-8080-exec-6
async threadpool executing
async threadpool name pool-3-thread-1
async threadpool Done
```

Configuration of ThreadPool

newCachedThreadPool

```
async Main thread name http-nio-8080-exec-91
async threadpool executing
async threadpool name pool-3-thread-92
async Main thread name http-nio-8080-exec-92
async threadpool executing
async threadpool name pool-3-thread-93
async Main thread name http-nio-8080-exec-93
async threadpool executing
async threadpool name pool-3-thread-94
async Main thread name http-nio-8080-exec-94
async threadpool executing
async threadpool name pool-3-thread-95
async Main thread name http-nio-8080-exec-95
async threadpool executing
async threadpool name pool-3-thread-96
async Main thread name http-nio-8080-exec-96
async threadpool executing
async threadpool name pool-3-thread-97
async Main thread name http-nio-8080-exec-97
async threadpool executing
async threadpool name pool-3-thread-98
async Main thread name http-nio-8080-exec-98
async threadpool executing
async threadpool name pool-3-thread-99
async Main thread name http-nio-8080-exec-99
async threadpool executing
async threadpool name pool-3-thread-100
async Main thread name http-nio-8080-exec-100
async threadpool executing
async threadpool name pool-3-thread-101
async threadpool Done
```

Configuration of ThreadPool

newSingleThreadExecutor

```
async threadpool name pool-3-thread-1
2024-12-22T23:47:39.436+08:00  INFO 11591 --- [pool-3-thread-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-22T23:47:39.757+08:00  INFO 11591 --- [pool-3-thread-1] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@3e45a991
2024-12-22T23:47:39.762+08:00  INFO 11591 --- [pool-3-thread-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
async threadpool Done
async Main thread name http-nio-8080-exec-2
async threadpool executing
async threadpool name pool-3-thread-1
async threadpool Done
completableFuture Main thread name http-nio-8080-exec-3
CompletableFuture pool main thread executing
CompletableFuture pool main thread name http-nio-8080-exec-3
CompletableFuture pool step 1 thread executing
CompletableFuture pool step 1 thread name pool-3-thread-1
CompletableFuture pool step 2 thread executing
CompletableFuture pool step 2 thread name pool-3-thread-1
completableFuture Main thread name http-nio-8080-exec-4
CompletableFuture pool main thread executing
CompletableFuture pool main thread name http-nio-8080-exec-4
CompletableFuture pool step 1 thread executing
CompletableFuture pool step 1 thread name pool-3-thread-1
CompletableFuture pool step 2 thread executing
CompletableFuture pool step 2 thread name pool-3-thread-1
```

Configuration of ThreadPool

newScheduledThreadPool

```
completableFuture Main thread name http-nio-8080-exec-6
CompletableFuture pool main thread executing
CompletableFuture pool main thread name http-nio-8080-exec-6
CompletableFuture pool step 1 thread executing
CompletableFuture pool step 1 thread name pool-3-thread-1
CompletableFuture pool step 2 thread executing
CompletableFuture pool step 2 thread name pool-3-thread-2
completableFuture Main thread name http-nio-8080-exec-4
CompletableFuture pool main thread executing
CompletableFuture pool main thread name http-nio-8080-exec-4
CompletableFuture pool step 1 thread executing
CompletableFuture pool step 1 thread name pool-3-thread-1
CompletableFuture pool step 2 thread executing
CompletableFuture pool step 2 thread name pool-3-thread-2
completableFuture Main thread name http-nio-8080-exec-7
CompletableFuture pool main thread executing
CompletableFuture pool main thread name http-nio-8080-exec-7
CompletableFuture pool step 1 thread executing
CompletableFuture pool step 1 thread name pool-3-thread-1
CompletableFuture pool step 2 thread executing
CompletableFuture pool step 2 thread name pool-3-thread-2
```

Configuration of ThreadPool

ThreadPoolExecutor

```
@Bean
public Executor taskExecutor() {
    return new ThreadPoolExecutor(
        corePoolSize: 1,  // core thread. always running
        1,  // max thread till the max one when queue up to limit
        60, TimeUnit.SECONDS,  // max thread alive tim
        new LinkedBlockingQueue<>( capacity: 10),  // queue limit
        new ThreadPoolExecutor.CallerRunsPolicy()  // refuse policy   You, A minute ago • Uncommitted changes
);
}
```

AbortPolicy： raise the exception RejectedExecutionException

CallerRunsPolicy： main thread to do

DiscardPolicy： discard it with out any action

DiscardOldestPolicy： remove the last task in queue and retry the latest one

Configuration of ThreadPool

ThreadPoolExecutor

```java
@Bean
public Executor taskExecutor() {
    return new ThreadPoolExecutor(
        corePoolSize: 1,  // core thread. always running
        1,  // max thread till the max one when queue up to limit
        60, TimeUnit.SECONDS,  // max thread alive tim
        new LinkedBlockingQueue<>( capacity: 10),  // queue limit
        new ThreadPoolExecutor.CallerRunsPolicy()  // refuse policy   You, A minute ago • Uncommitted changes
);
}
```

CallerRunsPolicy： main thread to do

```
async threadpool name http-nio-8080-exec-84
async threadpool executing
async threadpool executing
async threadpool name http-nio-8080-exec-47
async threadpool executing
async threadpool name http-nio-8080-exec-10
async threadpool executing
async threadpool name http-nio-8080-exec-102
async threadpool executing
async threadpool name http-nio-8080-exec-96
async threadpool executing
async threadpool name http-nio-8080-exec-109
async threadpool executing
async threadpool executing
async threadpool name http-nio-8080-exec-123
async threadpool name http-nio-8080-exec-94
async threadpool name http-nio-8080-exec-81
async threadpool executing
async threadpool name http-nio-8080-exec-119
async threadpool executing
async threadpool name http-nio-8080-exec-24
async threadpool executing
async threadpool name http-nio-8080-exec-38
async threadpool executing
async threadpool name http-nio-8080-exec-191
async threadpool executing
async threadpool executing
```

Configuration of ThreadPool

ThreadPoolExecutor

```
@Bean
public Executor taskExecutor() {
    return new ThreadPoolExecutor(
        corePoolSize: 1,  // core thread. always running
        1,  // max thread till the max one when queue up to limit
        60, TimeUnit.SECONDS,  // max thread alive tim
        new LinkedBlockingQueue<>( capacity: 10),  // queue limit
        new ThreadPoolExecutor.CallerRunsPolicy()  // refuse policy   You, A minute ago • Uncommitted changes
);
}
```

AbortPolicy： raise the exception RejectedExecutionException

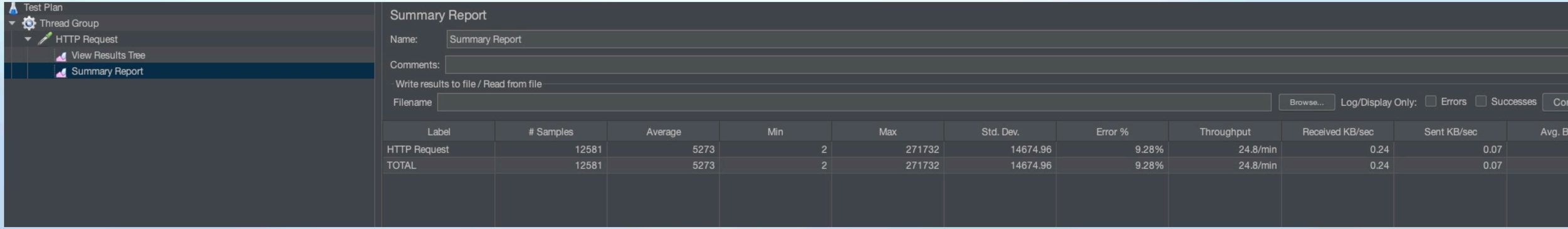
```
java.util.concurrent.RejectedExecutionException Create breakpoint : Task java.util.concurrent.CompletableFuture$AsyncSupply@3faea708 rejected from java.util.concurrent.ThreadPoolExecutor@28bb9f9d[Running, pool size = 1, active threads = 1, queued tasks = 10, completed tasks = 0] <3 interr
```

Configuration of ThreadPool

ThreadPoolExecutor

```
@Bean
public Executor taskExecutor() {
    return new ThreadPoolExecutor(
        corePoolSize: 1,  // core thread. always running
        1,  // max thread till the max one when queue up to limit
        60, TimeUnit.SECONDS,  // max thread alive tim
        new LinkedBlockingQueue<>( capacity: 10),  // queue limit
        new ThreadPoolExecutor.CallerRunsPolicy()  // refuse policy   You, A minute ago • Uncommitted changes
);
}
```

DiscardPolicy： discard it with out any action



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. B
HTTP Request	12581	5273	2	271732	14674.96	9.28%	24.8/min	0.24	0.07	
TOTAL	12581	5273	2	271732	14674.96	9.28%	24.8/min	0.24	0.07	

Configuration of ThreadPool

ThreadPoolExecutor

```
@Bean
public Executor taskExecutor() {
    return new ThreadPoolExecutor(
        corePoolSize: 1,  // core thread. always running
        1,  // max thread till the max one when queue up to limit
        60, TimeUnit.SECONDS,  // max thread alive tim
        new LinkedBlockingQueue<>( capacity: 10),  // queue limit
        new ThreadPoolExecutor.CallerRunsPolicy()  // refuse policy   You, A minute ago • Uncommitted changes
);
}
```

DiscardOldestPolicy： remove the last task in queue and retry the latest one

Configuration of ThreadPool

线程池类型	描述	适用场景
FixedThreadPool	固定大小的线程池，线程数量不变。	适用于负载固定的场景，如并发量稳定的任务。
CachedThreadPool	可根据需要创建线程，线程池中线程数量随任务量而动态增加。	适用于高并发、短期任务的场景，适合轻量级任务处理。
SingleThreadExecutor	只有一个线程来执行任务，保证任务按顺序执行。	适用于需要按顺序执行的任务，如日志处理、文件操作等。
ScheduledThreadPoolExecutor	支持定时任务和周期性任务的线程池。	适用于定时任务、周期性任务或延迟任务的调度。
ThreadPoolExecutor	可以高度自定义的线程池，允许设置核心线程数、最大线程数等。	适用于需要高度自定义线程池配置的复杂应用场景。

Configuration of ThreadPool

```
{
  "data": {
    "activeCount": 0,
    "queueSize": 0,
    "corePoolSize": 2,
    "completedTaskCount": 342,
    "maxPoolSize": 4
  },
  "code": "200"
}
```

```
FeHelper  排序: 默认 升序 降序  乱码修正

{
  "data": {
    "activeCount": 4,
    "queueSize": 10,
    "corePoolSize": 2,
    "completedTaskCount": 328,
    "maxPoolSize": 4
  },
  "code": "200"
}
```

```
no usages  new *
@Bean
public Executor taskExecutor() {
    return new ThreadPoolExecutor(
        corePoolSize: 1,  // core thread. always running
        1,  // max thread till the max one when queue up to limit
        60, TimeUnit.SECONDS,  // max thread alive tim
        new LinkedBlockingQueue<>( capacity: 10),  // queue limit
        new ThreadPoolExecutor.DiscardPolicy()  // refuse policy
);
}

no usages  new *
@Bean
public ThreadPoolTaskExecutor executor() {
    ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
    executor.setCorePoolSize(2);
    executor.setMaxPoolSize(4);
    executor.setQueueCapacity(10);
    executor.setThreadNamePrefix("springboot-thread-");
    executor.initialize();   You, 2 minutes ago • Uncommitted changes
    return executor;
}
```

```
no usages  new *
@RequestMapping(value = "/monitor",method = RequestMethod.GET)
public Response monitor()  {
    int activeCount = executor.getActiveCount();
    int corePoolSize = executor.getCorePoolSize();
    int maxPoolSize = executor.getMaxPoolSize();   You, 6 minutes ago • Uncommitted changes
    int queueSize = executor.getThreadPoolExecutor().getQueue().size();
    long completedTaskCount = executor.getThreadPoolExecutor().getCompletedTaskCount();
    HashMap data = new HashMap<String,Integer>();
    data.put("activeCount",activeCount);
    data.put("corePoolSize",corePoolSize);
    data.put("maxPoolSize",maxPoolSize);
    data.put("queueSize",queueSize);
    data.put("completedTaskCount",completedTaskCount);
    return Response.success(data);
}
}
```

Configuration of ThreadPool

ForkJoinPool - Computationally optimized

For algorithm

https://www.bilibili.com/video/BV1M34y1q7M2/?spm_id_from=333.337.search-card.all.click&vd_source=777b66d9ea6bb56ea53f120df4b32bb6

For impl

https://liaoxuefeng.com/books/java/threading/fork-join/index.html

Configuration of ThreadPool

SyncTaskExecutor

```java
@Bean(name = "synctaskExecutor")
public TaskExecutor synctaskExecutor() {
    return new SyncTaskExecutor();    You, A minute ago • Uncommitted changes
}
```

```java
@Override
@Async("synctaskExecutor")    You, A minute ago • Uncommitted changes
public CompletableFuture asyncService() {
    System.out.println("async threadpool executing");
    System.out.println("async threadpool name " + Thread.currentThread().getName());
    int insertResult = threadMapper.insertThreadLog( function: "asyncPool");
    System.out.println("async threadpool Done");
    return new CompletableFuture<String>().completedFuture( value: "threadpool insertResult:"+ insertResult);
}
```

```
async Main thread name http-nio-8080-exec-1
async threadpool executing
async threadpool name http-nio-8080-exec-1
2024-12-23T01:17:44.337+08:00  INFO 12754 --- [nio-8080-exec-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-12-23T01:17:44.705+08:00  INFO 12754 --- [nio-8080-exec-1] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@64f1fb04
2024-12-23T01:17:44.709+08:00  INFO 12754 --- [nio-8080-exec-1] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
async threadpool Done
async Main thread name http-nio-8080-exec-4
async threadpool executing
async threadpool name http-nio-8080-exec-4
async threadpool Done
```

Q&A