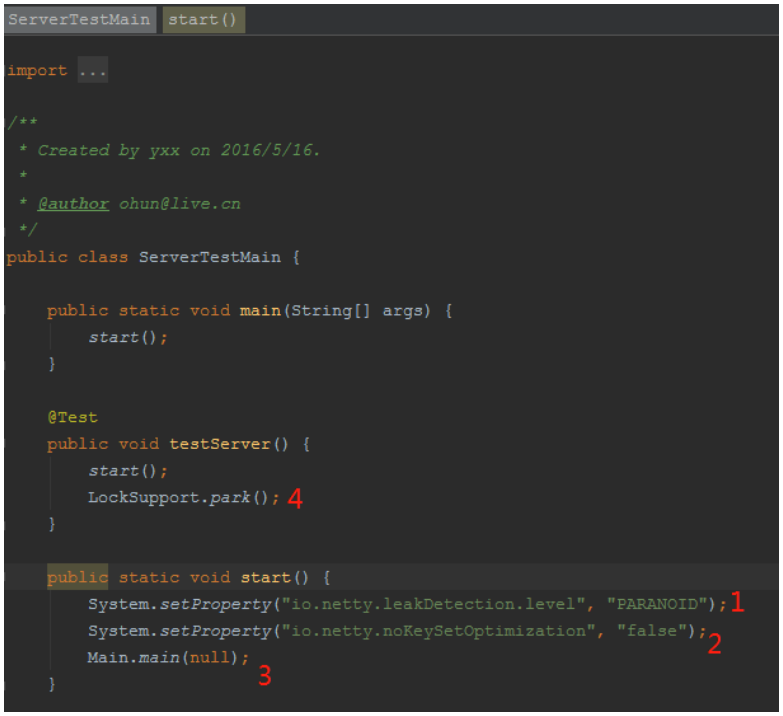


1、修改默认的内存泄露检测级别

netty中用到内存泄露检测的地方主要有：

*CompositeByteBuf；

*HashedWheelTimer；

*继承AbstractByteBufAllocator的几个类。

netty的内存泄露检测分为四级：

DISABLED: 不进行内存泄露的检测；

SIMPLE: 抽样检测，且只对部分方法调用进行记录，消耗较小，有泄漏时可能会延迟报告，默认级别；

ADVANCED: 抽样检测，记录对象最近几次的调用记录，有泄漏时可能会延迟报告；

PARANOID: 每次创建一个对象时都进行泄露检测，且会记录对象最近的详细调用记录。是比较激进的内存泄露检测级别，消耗最大，建议只在测试时使用。

如果需要修改默认的检测级别，可以通过：1、调用静态方法setLevel进行修改；2、设置启动参数io.netty.leakDetectionLevel。

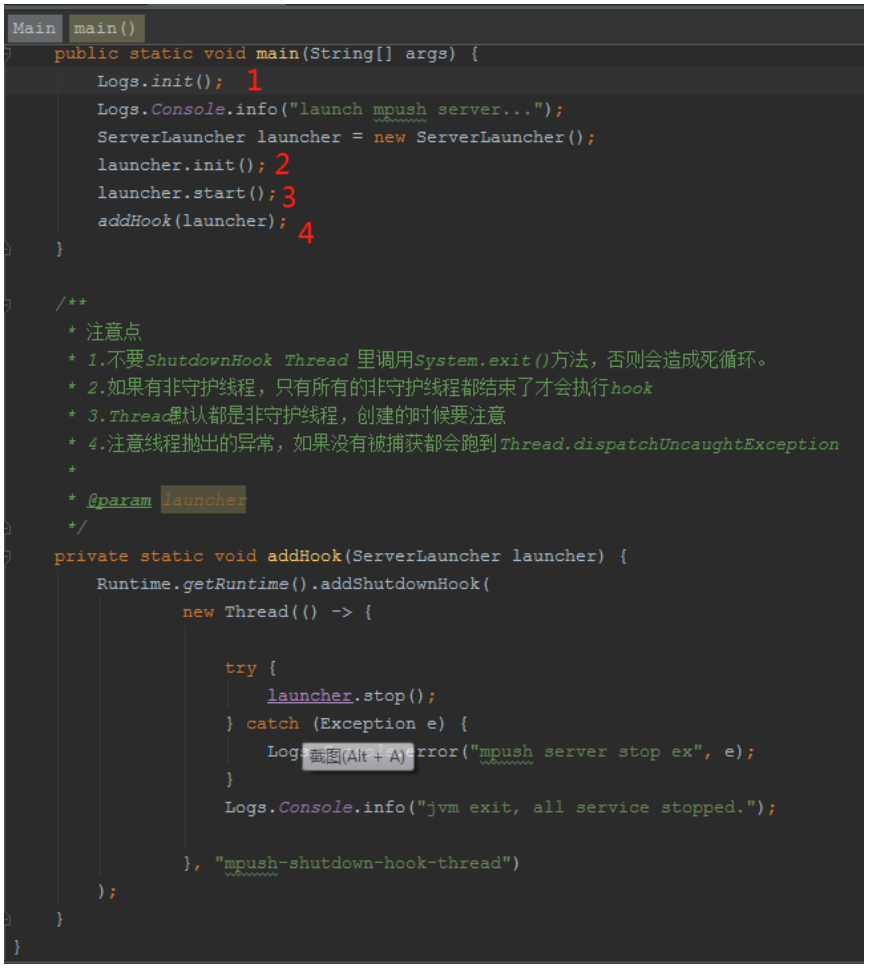
由于内存泄露主要是对某一类资源的检测，因此对于同一类的对象，只需实例化一个ResourceLeakDetector， 否则起不到检测的作用。

2、关闭selectedKeys优化功能

Netty对Selector的selectedKeys进行了优化，用户可以通过io.netty.noKeySetOptimization开关决定是否启用该优化项。默认不打开selectedKeys优化功能。

3、启动入口MAIN

4、阻塞主线程



1、各模块的日志对象初始化

2、服务初始化

3、服务启动

4、JVM钩子，JVM退出时调用

服务初始化

ServerLauncher#init()

```
ServerLauncher  init()
    private ServerEventListener serverEventListener;

    public void init() {
        if (mPushServer == null) {
            mPushServer = new MPushServer();  1
        }

        if (chain == null) {
            chain = BootChain.chain();  2
        }

        if (serverEventListener == null) {
            serverEventListener = ServerEventListenerFactory.create();  3
        }

        serverEventListener.init(mPushServer); 4

        chain.boot()  5
                .setNext(new CacheManagerBoot())//1.初始化缓存模块
                .setNext(new ServiceRegistryBoot())//2.启动服务注册与发现模块
                .setNext(new ServiceDiscoveryBoot())//2.启动服务注册与发现模块
                .setNext(new ServerBoot(mPushServer.getConnectionServer(), mPushServer.getConnServerNode()))//3.启动接入服务
                .setNext(() -> new ServerBoot(mPushServer.getWebsocketServer(), mPushServer.getWebsocketServerNode()), wsEnabled())//4.启动websocket接入服务
                .setNext(() -> new ServerBoot(mPushServer.getUdpGatewayServer(), mPushServer.getGatewayServerNode()), udpGateway())//5.启动udp网关服务
                .setNext(() -> new ServerBoot(mPushServer.getGatewayServer(), mPushServer.getGatewayServerNode()), tcpGateway())//6.启动tcp网关服务
                .setNext(new ServerBoot(mPushServer.getAdminServer(), null))//7.启动控制台服务
                .setNext(new RouterCenterBoot(mPushServer))//8.启动路由中心组件
                .setNext(new PushCenterBoot(mPushServer))//9.启动推送中心组件
                .setNext(() -> new HttpProxyBoot(mPushServer), CC.mp.http.proxy_enabled)//10.启动http代理服务，dns解析服务
                .setNext(new MonitorBoot(mPushServer))//11.启动监控服务
                .end();
    }
```

1、MPushServer初始化

2、实例化启动链对象BootChain(链表结构)

3、通过SPI，获取mpush-core中服务ServerEventListenerFactory的实例

DefaultServerEventListener

4、初始化（我也不知道干嘛的）

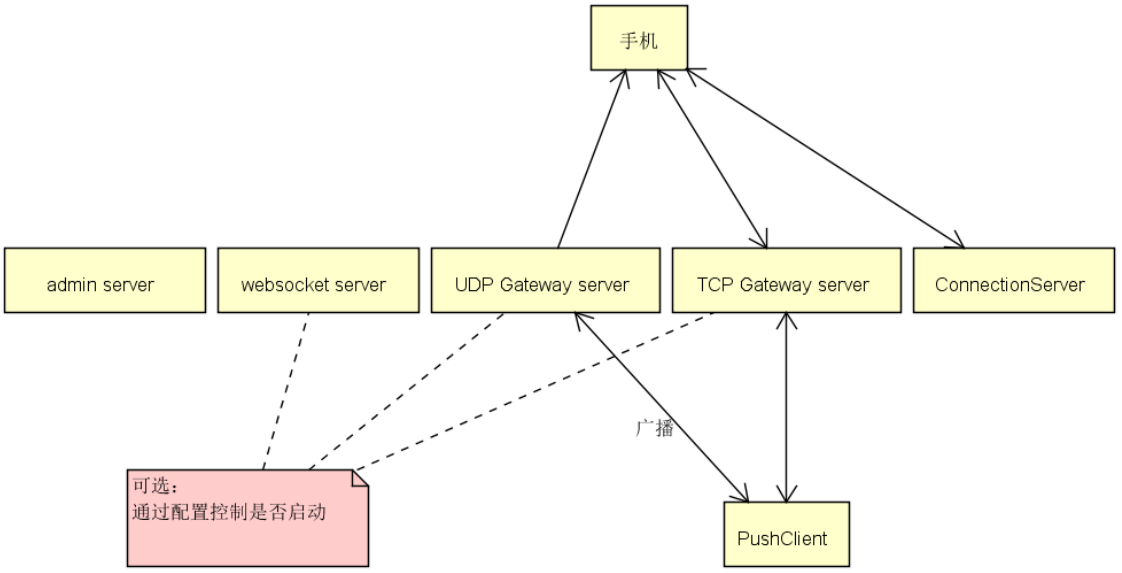
5、初始化一系列模块，并加入到BootChain链表中

- 初始化缓存模块
- 服务注册模块
- 服务发现模块
- 接入服务
- websocket接入服务
- udp网关服务
- tcp网关服务
- 控制台服务
- 路由中心组件
- 推送中心组件
- http代理服务，dns解析服务
- 监控服务

服务启动

ServerLauncher#start()

链式调用以上各个模块中的start()方法；


手机
admin server
websocket server
UDP Gateway server
TCP Gateway server
ConnectionServer
广播
可选：
通过配置控制是否启动
PushClient