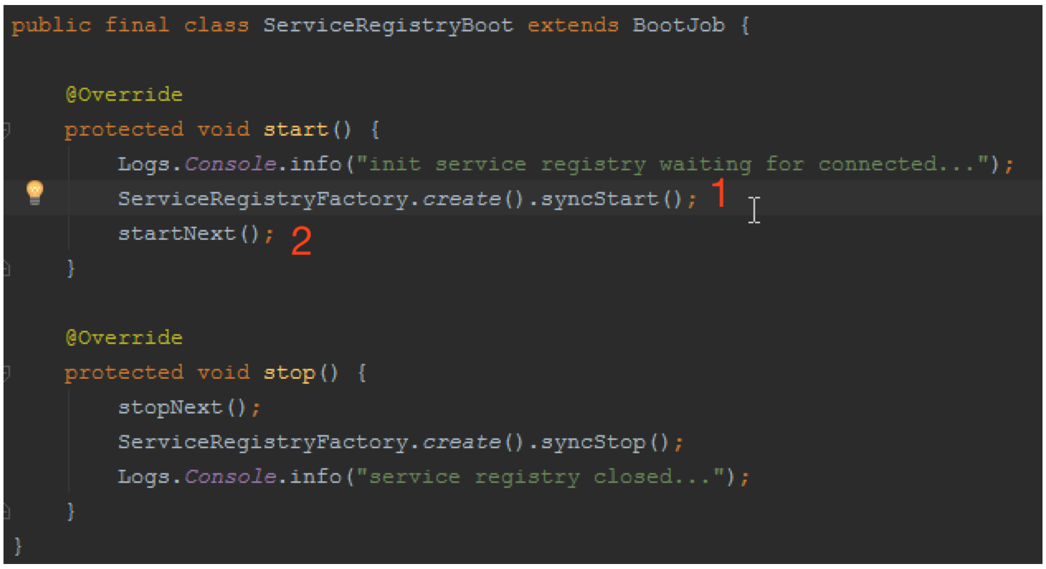
服务注册模块

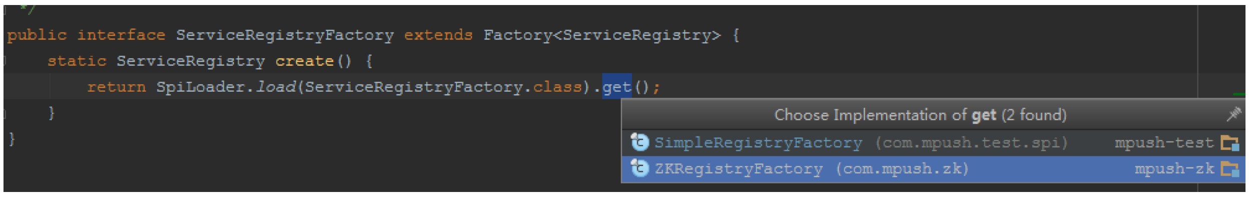


```java
public final class ServiceRegistryBoot extends BootJob {

    @Override
    protected void start() {
        Logs.Console.info("init service registry waiting for connected...");
        ServiceRegistryFactory.create().syncStart(); 1
        startNext(); 2
    }

    @Override
    protected void stop() {
        stopNext();
        ServiceRegistryFactory.create().syncStop();
        Logs.Console.info("service registry closed...");
    }
}
```

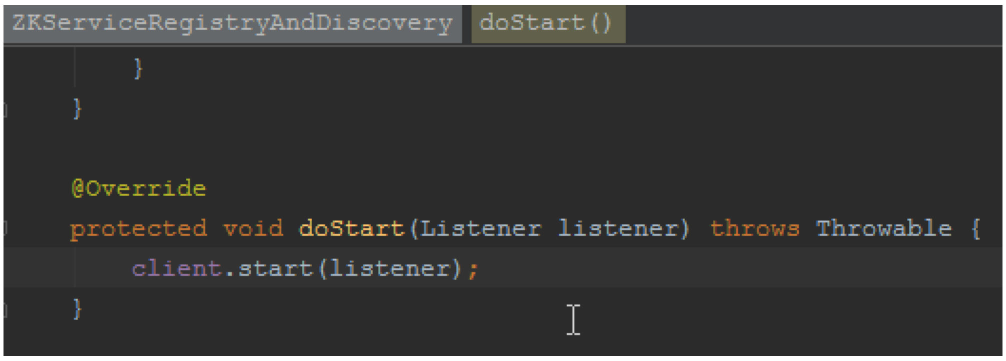
1、通过SPI，找到mpush-zk模块中ServiceRegistryFactory接口的实现类ZKRegistryFactory，得到ZKServiceRegistryAndDiscovery实例；



```java
public interface ServiceRegistryFactory extends Factory<ServiceRegistry> {
    static ServiceRegistry create() {
        return SpiLoader.load(ServiceRegistryFactory.class).get();
    }
}
```

```
Choose Implementation of get (2 found)
SimpleRegistryFactory (com.mpush.test.spi)    mpush-test
ZKRegistryFactory (com.mpush.zk)              mpush-zk
```

先调用ZKServiceRegistryAndDiscovery的父类BaseService#syncStart()方法，进而调用ZKServiceRegistryAndDiscovery#doStart()启动ZKClient；



```java
ZKServiceRegistryAndDiscovery  doStart()
        }
    }

    @Override
    protected void doStart(Listener listener) throws Throwable {
        client.start(listener);
    }
```

2、调用startNext继续调用下一个BootChain

服务发现模块

与上面类似，最终也是通过SPI，找到mpush-zk模块中ServiceDiscoveryFactory接口的实现类ZKDiscoveryFactory，得到ZKServiceRegistryAndDiscovery实例；

总结：

这两个过程，都是拿到同一个ZKServiceRegistryAndDiscovery实例，然后调用同一个doStart()启动ZKClient，不会启动ZKClient两次，因为做了Listener控制(见BaseService)