代码检查

检查日期20221209，检查人：朱立俊

代码量统计

开发人员	ID	变化	代码行数	平均变化代码行数
马天浩	s_math	23174 (50.1%)	40371 (54.2%)	1.7
曹红亮	s_caohl	22869 (49.5%)	28722 (38.6%)	1.2
沈永康	s_shenyk	71 (0.2%)	2360 (3.2%)	33.2
王浩南	s_wanghn	45 (0.1%)	1063 (1.4%)	23.6
蒋易恒	s_jiangyh	29 (0.1%)	862 (1.2%)	29.7
石昊	s_shihao	13 (0.0%)	584 (0.8%)	44.9
沈建树	s_shenjs	11 (0.0%)	489 (0.7%)	44.4

阻断错误

1. 包名错误 责任人：沈永康

```
com.pde.pdes.portal.chronicle.service.Impl
```

2. 函数名称重复 责任人：沈永康

```
# chronicle.js
export function deleteChronicleByIds(ids)
```

代码不规范

1. if/else/while/do语句必须使用{} 责任人：马天浩

```
public class MyBeanUtilsHelper {
    ...
    if (srcValue == null) emptyNames.add(pd.getName());
    ...
}
```

2. 应以impl结尾 责任人：曹洪亮

```
public class HonorServiceImp extends ServiceImpl<HonorMapper, HonorPO> implements
```

3. 包名应都为小写 责任人：王浩南

```
com.pde.pdes.portal.standard.Utils
```

注释不规范

责任人：所有 注释应符合javadoc规范，类和函数必须有注释，并满足javadoc规范，可参考以下代码

```
/**
 * 属性读取配置类
 *
 * @author wzheng
 * @version 1.0.0
*/
@Configuration
public class PdeCoreConfig {

    /**
    * @Fields corePoolSize : 设置核心线程数
    */
    @Value("${pde.core.PdeDefaultThreadPool.CorePoolSize:30}")
    private int corePoolSize = 30;

    /**
    * @Fields maxPoolSize : 设置最大线程数
    */
    @Value("${pde.core.PdeDefaultThreadPool.MaxPoolSize:150}")
    private int maxPoolSize = 150;

    /**
    * @Fields queueCapacity : 设置队列容量
    */
    @Value("${pde.core.PdeDefaultThreadPool.QueueCapacity:100}")
    private int queueCapacity = 100;

    /**
    * @Fields keepAliveSeconds : 设置线程活跃时间（秒）
    */
    @Value("${pde.core.PdeDefaultThreadPool.KeepAliveSeconds:60}")
    private int keepAliveSeconds = 60;

    /**
    * @Fields threadNamePrefix : 线程名前缀
    */
    @Value("${pde.core.PdeDefaultThreadPool.ThreadNamePrefix:pde-thread-}")
    private String threadNamePrefix = "pde-thread-";
```

```
    /**
    * 获取线程池
    *
    * @return 线程池
    */
    @Bean(name = "PdeDefaultThreadPool")
    public ThreadPoolTaskExecutor getExecutor() {
        ThreadPoolTaskExecutor taskExecutor = new ThreadPoolTaskExecutor();
        taskExecutor.setCorePoolSize(this.corePoolSize);
        taskExecutor.setMaxPoolSize(this.maxPoolSize);
        taskExecutor.setQueueCapacity(this.queueCapacity);
        taskExecutor.setKeepAliveSeconds(this.keepAliveSeconds);
        taskExecutor.setThreadNamePrefix(this.threadNamePrefix);
        // 设置拒绝策略
        taskExecutor.setRejectedExecutionHandler(new
ThreadPoolExecutor.CallerRunsPolicy());
        // 等待所有任务结束后再关闭线程池
        taskExecutor.initialize();
        return taskExecutor;
    }
}
```