# Leb 用户手册

# 问题引出

某企业S有一个企业内部系统（简称SI），SI是20多人的团队协作开发的，设计之初进行合理的模块划分，如下图所示（仅仅是举个例子，并不严谨）。

项目目录结构

其中每一项都是maven工程，箭头代表的是工程目录之间的包含关系，并不代表依赖关系。

需求文档描述，要求只有注册用户才可以登录系统，要求所有的资源（如：报表）均包含创建人、创建时间、修改人，修改时间等日志型字段，以便跟踪记录，系统中有很多的资源，如Report是一种资源，其他资源未画出，资源在运行的时候需要通过并发License计数，资源在编辑的时候需要加并发锁。

问题一：

当登录用户点击注销按钮时，需要清理该用户登录过程中占用的所有资源，包括如下：

1. 释放登录所占应的并发license；
2. 释放用户登录期间运行的所有资源并发license；
3. 释放用户登录期间编辑的所有资源并发锁；
4. 清理用户产生的临时文件临时表等；
5. 删除用户的运行期上下文。

License和Lock的Domain在SI-Framework中，资源的Domain在每个资源工程中。User的Domain在SI-User工程中，并且各种资源之间还有相互依赖关系，这导致整个工程依赖链非常复杂。

以SI-Framework、SI-User和SI-Report工程为例。SI-User和SI-Report必须依赖SI-Framework，同时SI-Report必须依赖SI-User，SI-Framework要反过来调用SI-Report以便释放每种资源请求的license和lock。怎么解决这种循环依赖呢，目前的做法是SI-Framework中增加接口，实现放到各个资源工程去，通过Spring容器查找所有实现类，迭代调用。

两年后，由于系统越做越大，变得模块众多，依赖混乱不堪，到处弥漫着臭味。

请思考需求中的如下两个问题：

1. 当在用户管理中永久删除一个用户时候要求同时删除该用户创建的所有资源，否则该用户的一些数据永远删不掉，比如个人文件夹中的信息，比如对资源的授权信息等，这部分数据成了垃圾信息。这个需求该怎么实现？
2. 理论上讲每个工程都可以实现注入其他工程的Service或Domain以便调用，但是我们往往只是需要 调用Service中的某一个方法，为什么要依赖整个Service呢，为什么要全部暴露？

问题二：在系统上线初期，报表是在用户访问时直接查询数据仓库，实时计算生成的，用户之间不能共享计算过程，即便同一个用户多次访问也不能共享，这导致系统非常耗资源且慢，用户体验极差，后来增加了缓存和快照解决了问题。这里只谈快照，快照指的是对于一部分报表，系统每晚凌晨自动调度生成结果保存好，这个结果成为快照，第二天用户查看报表时。直接将快照返回。

当快照生成时，系统需要做如下动作：

1. 系统到设定时间后调度生成报表快照（也可以手动触发）；
2. 存储生成的快照到个人文件夹和公共文件夹；
3. 保存日志信息，无论是否出错均保存；
4. 发邮件给管理员或调度配置人员；
5. 发消息到调度配置人员的个人中心；

这里的3)、4)、5)步是可以并发执行的，完全没有必要按顺序执行。（当然如果是系统自动触发的调度，对实时性没有要求，不并发也问题不大，我只是举个例子）

上面这些问题怎么解决呢？Event Bus！

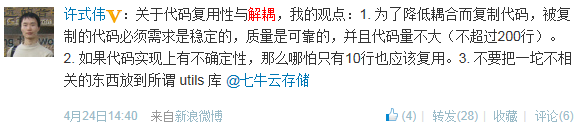
# 事件总线软件

# 软件终极目标

软件产品的终极目标-松耦合，他们的出现都是为解决耦合问题：IoC，AOP，SOA，EDA，观察者模式，面向接口编程，MVC。。。

GO语言

<http://www.oschina.net/news/34381/go-in-google>



补充：去《敏捷》和《unix编程艺术》《程序员修炼之道》找相关章节

# 事件总线

事件驱动架构EDA（Event-Driven Architecture）。

本质：事件总线（event bus）的本质是观察者模式。

优点：解耦，依赖反转，反转控制，好莱坞原则；高性能，天生多线程；易维护，随时增加修改；符合自然思维，一件事情发生引发另一件事。

# Leb 快速入门

下面我们写个HelloWorld，Say.java是一个普通的POJO，并在相应的监听方法上加上@Observe注解。将Say的实例注册到Lebs中，之后便可以通过总线发送消息，相应的@Observe便会处理。

|  |
| --- |
| **public** **class** Say {  @Observe(hint = "say")  **public** **void** sayHello() {  Logs.*p*("hello leb");  }  @Observe(hint = "say")  **public** **void** sayTo(String who) {  Logs.*p*("hello " + who);  }  @Observe  **public** **void** bye() {  Logs.*p*("bye leb");  }  @Observe(hint = "say")  **public** **void** sayTime(Date date) {  Logs.*p*(Times.*format*(date));  }  } |
| Say.java |

|  |
| --- |
| Lebs leb = Lebs.*singleton*();  leb.observers(**new** Say());  leb.notice("say");// hello leb  leb.notice("say", "bruce");// hello bruce  leb.notice("bye");// bye leb  leb.notice("say", **new** Date());// 2013-05-19 13:53:16.016  leb.broadcast("say", 1);// 不执行任何方法，没有匹配上的参数 |
| Test.java |

# Leb架构与接口



Lebs

# FAQ

# 附录

# 更新日志

# 0.0.1发行注记

# 我的联系方式

|  |  |
| --- | --- |
| W | A |
| Weibo | @i不如 |
| Weixin | bruce\_sha |
| QQ | 258274473 |
| Email | [lbs.sha@gmail.com](mailto:lbs.sha@gmail.com) |
| GitHub | <http://github.com/bruce-sha> |

# 参考文献