Atitit 安全领域技术研究

# 身份验证cookie session token

加密解密aes rsa非对称

Md5签名

https

Url重定向隐藏

盾击模式

熔断器（cpu 内存 磁盘空间安全）

Cpu安全 熔断器 死锁问题

内存泄漏

连接泄漏 文件句柄泄漏

# 语言安全性比较

## [Type Safety](https://link.zhihu.com/?target=http://en.wikipedia.org/wiki/Type_safety" \t "_blank)，[Memory Safety](https://link.zhihu.com/?target=http://en.wikipedia.org/wiki/Memory_safety" \t "_blank) Gc

所谓语言安全性在于几个方面，例如[Type Safety](https://link.zhihu.com/?target=http://en.wikipedia.org/wiki/Type_safety" \t "_blank)，[Memory Safety](https://link.zhihu.com/?target=http://en.wikipedia.org/wiki/Memory_safety" \t "_blank)等等。这些Safety主要就是帮助程序员写出稳定的程序（如Type Safety），还有让程序的错误不要蔓延出去（如Memory Safety）等等。你看过这两个链接就应该知道为什么说Java比C/C++安全了。

## 异常处理机制 错误恢复机制

## 资源安全加载机制

们在做手机应用的时候，90%的客户端崩溃是由于资源问题导致的

比如有个图片文件找不到

而手机用户流失最大的原因之一就是客户端崩溃问题。

所以我多次问及客户端工程师，为什么不能将资源方面的异常捕捉到，用通用图片之类代替之类的方案来保证客户端的稳定性？

客户端工程师给我的答案是，问题就要暴露出来，才有助于改进。

而java是服务器开发居多，java最基础的要求之一就是 要掌握try-catch，对于非灾难性和非不可解决的问题的时候，留下日志，服务器得接着干活。

这里我解释下这种思维区别的不完整原因

第一个是服务器是对成百上千的用户负责，一个用户产生的异常行为不应该影响到其他用户。

而客户端基本只对当前使用用户负责，所以出了错不会影响到其他用户的感知。

其次呢，对于资源的加载，服务器往往是启动加载，也就是说在启动的时候加载，所以问题暴露比较早，就能比较及时的解决，错误日志也都在服务器本地。

而客户端呢，资源错误发生时机往往和业务逻辑错误相似，就是使用时暴露，相对来就比较难提前发现。而在使用中出现异常后，错误日志是需要发送到服务器的，所以客户端工程师往往不能最准时获得问题的详细情况。

所以最后的结果是，使用java的人仿佛总能提前发现问题，而c++却只能等客户端发错误报告来

## 集合边界范围检查

Ref

Atitit 安全领域 身份验证技术总结