Atitit 提升稳定性 错误处理 全局错误捕获

[1.2. 可以uncaughtException来全局捕获未捕获的Error， 使用uncaughtException 2](#_Toc19505)

[1.2.1. 使用 try/catch 3](#_Toc26170)

[1.3. 守护NodeJS进程和记录错误日志 3](#_Toc27314)

[1.3.1. 使用node来守护node 4](#_Toc21699)

[1.3.2. 使用shell启动脚本守护node 4](#_Toc8852)

[2. 参考资料 6](#_Toc32201)

[2.1. Atitit.android崩溃日志 全局异常捕获机制.docx 6](#_Toc9248)

[2.2. atitit js浏览器环境下的全局异常捕获　v2 qa1 (2).docx 6](#_Toc14186)

[2.3. C# WinForm捕获全局异常 - 跟着阿笨一起玩.NET - 博客园.mhtml 6](#_Toc8207)

/users/myinvation/5a96142c4431fe1878f66cf0/20276a9d1016c164cf840f3e3dd935f9

/users/userinfo/5a96142c4431fe1878f66cf0/20276a9d1016c164cf840f3e3dd935f9

##### **1、callback function**

通过异步回调来处理异步操作可能是目前nodejs中使用最广泛的方案。通过约定回调函数中第一个参数，为error对象，来传递错误，当第一个参数为null时表示没有错误。为什么要这么做呢，因为异步中的错误是无法通过try catch捕获的。

2、我们上面的例子中演示的，不管采用那种方式，我们都不能处理异步中抛出的异常，异步中的异常只能通过传递来传递给调用者，所以，在异步中应该尽可能的少做逻辑，只是作为必要的操作等，比如，我们可以通过异步来获取数据，但数据的格式化等，则放到异步之外（也就是上面演示中的setTimeout之外），这样的话，绝大部分的异常都能被捕获，应用crash的几率也会大大的减少。

最后，尽管我们能捕获大部分错误，但是我们还是应该把捕获到的异常进行记录分析，尽可能减少异常，并且，通过完善的单元测试来事先发现问题。

#### **异步异常的特点**

由于node的回调异步特性，无法通过try catch来捕捉所有的异常：

try {

process.nextTick(function () {

foo.bar();

});} catch (err) {

//can not catch it}

#### **domain**

在node v0.8+版本的时候，发布了一个模块domain。这个模块做的就是try catch所无法做到的：捕捉异步回调中出现的异常。

## 可以uncaughtException来全局捕获未捕获的Error， **使用uncaughtException**

我们可以uncaughtException来全局捕获未捕获的Error，同时你还可以将此函数的调用栈打印出来，捕获之后可以有效防止node进程退出，如：

process.on('uncaughtException', function (err) {  
 //打印出错误  
 console.log(err);  
 //打印出错误的调用栈方便调试  
 console.log(err.stack)；  
});

这相当于在node进程内部进行守护， 但这种方法很多人都是不提倡的，说明你还不能完全掌控Node.JS的异常。

### **使用 try/catch**

我们还可以在回调前加try/catch，同样确保线程的安全。

## **守护NodeJS进程和记录错误日志**

现在已经基本上解决了Node.JS因异常而崩溃的问题，不过任何平台都不是100%可靠的，还有一些错误是从Node底层抛出的，有些异常try/catch和uncaughtException都无法捕获。之前在运行ourjs的时侯，会偶尔碰到底层抛出的文件流读取异常，这就是一个底层libuv的BUG，node.js在0.10.21中进行了修复。  
  
面对这种情况，我们就应该为nodejs应用添加守护进程，让NodeJS遭遇异常崩溃以后能马上复活。  
  
另外，还应该把这些产生的异常记录到日志中，并让异常永远不再发生。

### **使用node来守护node**

[node-forever](https://github.com/nodejitsu/forever" \t "http://ourjs.com/detail/_blank) 提供了守护的功能和LOG日志记录功能

### **使用shell启动脚本守护node**

使用node来守护的话资源开销可能会有点大，而且也会略显复杂，OurJS直接在开机启动脚本来进程线程守护。  
  
如在debian中放置的 ourjs 开机启动文件： [/etc/init.d/ourjs](https://github.com/newghost/ourjs/blob/master/tools/ourjs" \t "http://ourjs.com/detail/_blank)  
  
这个文件非常简单，只有启动的选项，守护的核心功能是由一个无限循环 while true; 来实现的，为了防止过于密集的错误阻塞进程，每次错误后间隔1秒重启服务

*//ati* process.on(**'uncaughtException'**, **function** (err) {  
 ***console***.log(**"\*\*\*\*\*\*\*\*err ati start"**);  
 *//打印出错误* ***console***.log(err);  
 *//打印出错误的调用栈方便调试* ***console***.log(err.**stack**);  
 ***console***.log(**"\*\*\*\*\*\*\*\*err ati end"**);  
});  
  
 *setTimeout*(**function** () {  
 **throw "err2 from timeout"**;  
 },15000);  
*setTimeout*(**function** () {  
 *// throw "err from timeout aft 5 sec";*},5000);  
***console***.log(**" time out aft"**);  
*//atiend*

如果fsr open page then timeouterr...can resume.....

If fst err time ot ,,then channel close...cant resume..

# 参考资料

## Atitit.android崩溃日志 全局异常捕获机制.docx

## atitit js浏览器环境下的全局异常捕获　v2 qa1 (2).docx

## C# WinForm捕获全局异常 - 跟着阿笨一起玩.NET - 博客园.mhtml