Casperjs

20151124

# CasperJS缓存相关

## PhantomJS缓存：

### 使用Command-line选项：

方法一：

需要配合以下选项：

--disk-cache=[true|false]

--local-storage-path=/some/path //本地缓存存放路径

--max-disk-cache-size=size //本地缓存上限，以KB为单位

方法二：

可以将需要的配置一JSON的格式存放到配置文件中：

{

/\* Same as: --disk-cache \*/

"diskacheEnabled": true,

/\* Same as: --max-disk-cache-size=1000 \*/

"maxDiskCacheSize": 1000,

/\* etc. \*/

}

更多：<http://phantomjs.org/api/command-line.html>

## 测试结果：

当开启--disk-cache选项时：

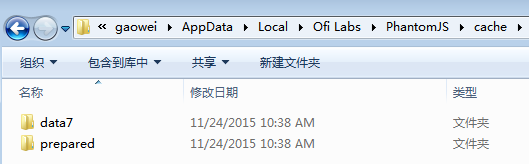


默认情况下在会在下面目录中创建两个子目录：

%LOCALAPPDATA%/Ofi Labs\PhantomJS\cache

两个子目录：data7 和prepared

如图：



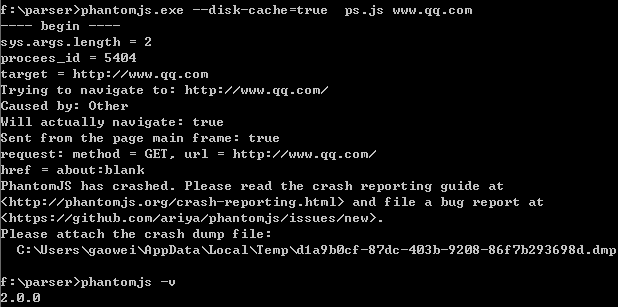
cache 同级目录下有对相关数据：



## 存在问题

通过参数修改存放路径无效。

改用2.0.0版本发现crash：



手动配置内部存储路径bug目前还没正式发布：

<https://github.com/ariya/phantomjs/issues/13689>

# CasperJS 异常退出

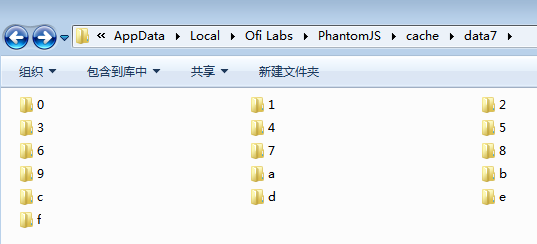
## 运行

直接带参数启动会导致PhantomJS carsh.

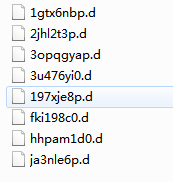


PhantomJS 版本为1.9.8

改用2.0.0版本能正常运行。



文件夹0中内容：



## 扫描缓存测试

### 测试点：

连续两次扫描，看是否因为前一次的缓存减少第二次爬取过程中的发包数量。

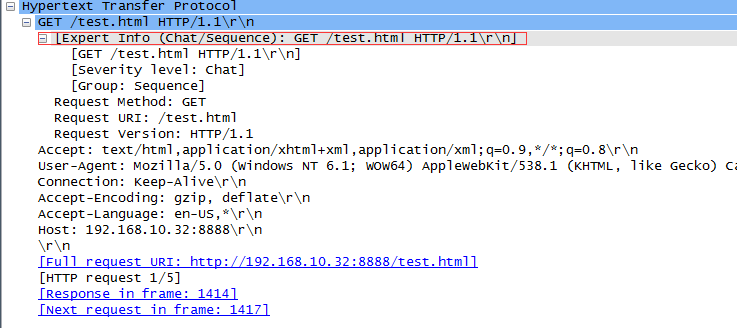
### 测试网站：

http://192.168.10.32:8888/test.html

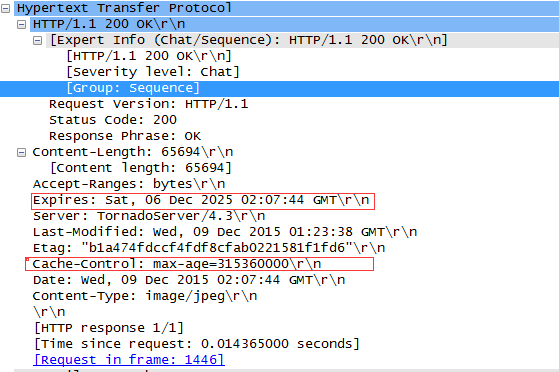
**过滤条件：**IMG_256

**设置的响应头部**：

1.页面的响应头部：（没有设置缓存）

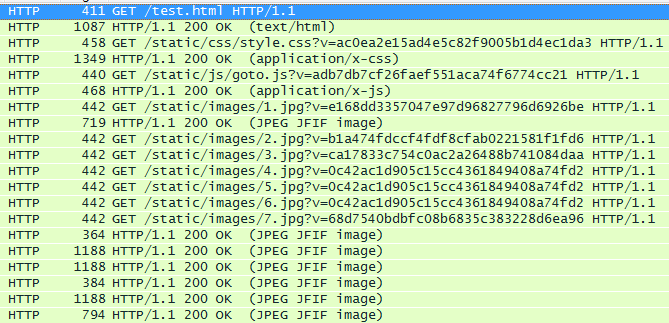


资源的响应头部：（设置了十年的缓存）



### 配置缓存后第一次抓取：

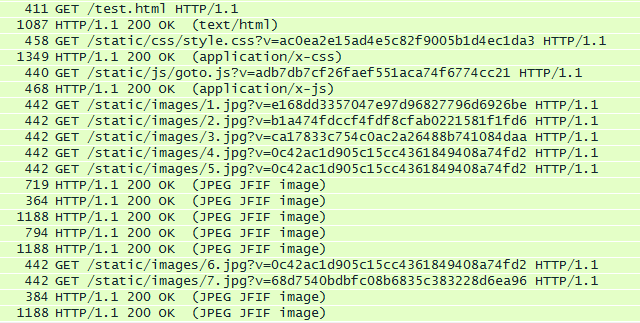
排除图片加载后抓取到的http请求：



## 配置缓存后第二次抓取



**2.4.不配置缓存后经行抓取：**



## 结论：

2.2-2.4的证据表明：在添加相应的选项后，phantomjs(casperjs)是支持缓存的

# 扫描部分网站卡死

## 现象

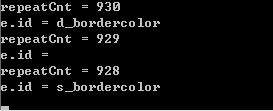
扫描网站如下网站卡死，phantomjs进程无法退出。

<http://192.168.24.190/ewebeditor/dialog/img.htm>

## 分析

跟踪定位发现调用showModalDialog接口：

找到对应的元素：



查看代码



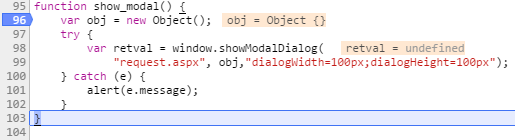


在chrome下showModalDialog已经在版本37时标明将陆续放弃支持，并在43版本正式停止支持。

参考：

<https://blog.chromium.org/2014/07/disabling-showmodaldialog.html>

<https://developer.mozilla.org/en-US/docs/Web/API/Window/showModalDialog>





## 解决方案：

采用timeout来处理 click事件：

casper.waitFor(function check() {

click('[crawl\_mark="targeted"]');

return true;

}, null, null, 500, null);

## 相关链接

<https://github.com/ariya/phantomjs/issues/10172>

# post数据无法获取

## 现象

post数据无法获取

## 分析

目前该功能在最新版本中能支持：

<https://github.com/ariya/phantomjs/issues/10158>

同时有个walkaround，使用内嵌的webserver来解决，对方法是post的请求特殊处理。可以参考：

<http://stackoverflow.com/questions/19550582/parsing-post-data-in-phantomjs>

# 获取phantomjs的进程id

## 需求：

在casperjs启动phantomjs进程时，获取到phantomjs进程的id

## 方案：

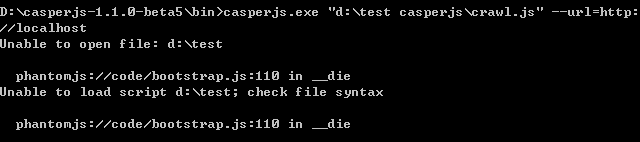
使用system模块：

var system = require('system');

casper.echo("phantomjs\_pid = " + system.pid);

# 脚本路径存在空格问题

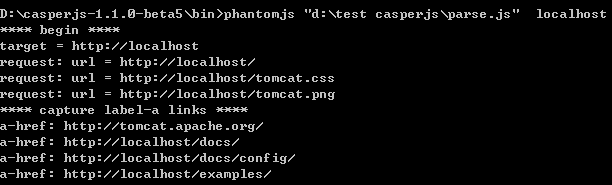
## 现象



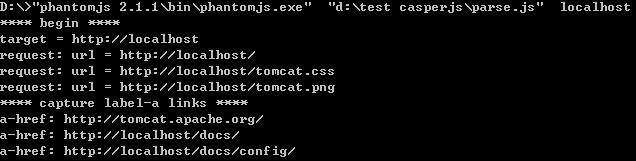
## 分析

Phantomjs用2.1.1版本时，路径没有问题：

脚本路径存在空格：



程序路径存在空格：



## 定位

之前我们已经知道phantomjs 2.1.1 版本没什么问题，所以先从casperjs中的bootstrap.js开始查看，发现脚本的最开的闭包函数中的system.args 就已经对空格进行断开，所以就从casperjs.exe中查看是否有问题。于是我们在casperjs.cs的源码中看到创建进程之前的一个步骤。

CASPER\_COMMAND.AddRange(new [] {

@"""" + Path.Combine(

Path.Combine(CASPER\_PATH, "bin"), "bootstrap.js") + @"""",

@"--casper-path=""" + CASPER\_PATH + @"""",

"--cli"

});

CASPER\_COMMAND.AddRange(CASPER\_ARGS);

ProcessStartInfo psi = new ProcessStartInfo();

psi.FileName = ENGINE\_FILE;

psi.UseShellExecute = false;

psi.RedirectStandardOutput = true;

psi.Arguments = String.Join(" ", CASPER\_COMMAND.ToArray());

try {

Process p = Process.Start(psi);

发现在拼接参数的是时候，存在隐患：

CASPER\_COMMAND.AddRange(CASPER\_ARGS);

这个代码中CASPER\_ARGS中的字串如果存在空格，在后续的拼接中并不会加上引号，所以在折行代码前加上单独的处理代码，如果字串中有空格，而且开头不是以引号开始，就在字串两端加上空格。这样处理也许后续会有问题，但至少能解决当下的问题：

for (int i = 0; i < CASPER\_ARGS.Count; i++) {

if (CASPER\_ARGS[i].Contains(" ") &&

!CASPER\_ARGS[i].StartsWith("\"")) {

CASPER\_ARGS[i]="\""+ CASPER\_ARGS[i] + "\"";

}

}

测试发现能解决问题。

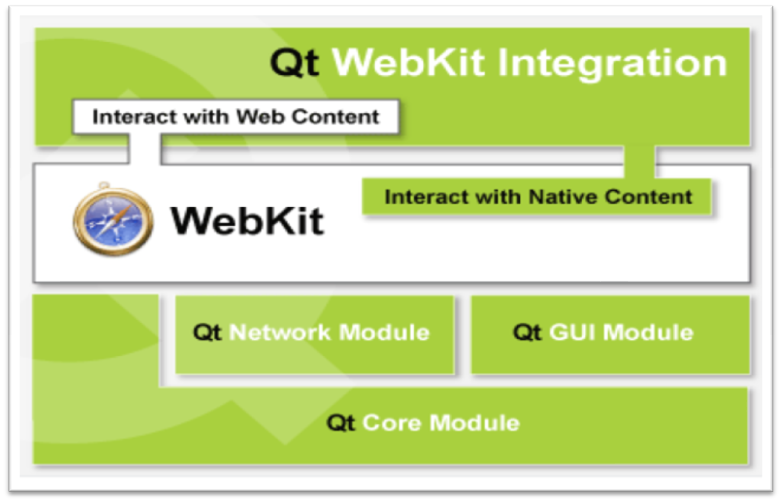
# QtWebKit学习

## 浏览器组件

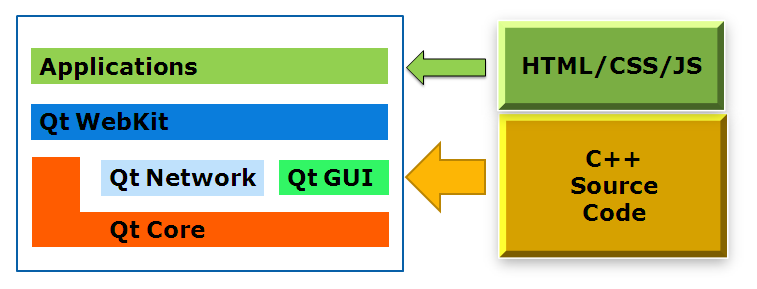
一个浏览器通常都有如下的标准组件：HTML、XML、CSS、JavsScript解析器，Layout，文字和图形渲染，图像解码，GPU交互，网络访问，硬件加速。

QtWebKit只是WebKit的其中一种port。

## QtWebKit架构图



对应的代码:



## WebKit

WebKit 是渲染引擎，JS引擎一般不属于WebKit，比如chrome中的JS引擎是V8。最开是是KDE开发KHTML后来APPLE参与，并将其改为WebKit。渲染引擎不只是提供单个页面渲染，还包括工作流程控制相关内容。

<http://www.infoq.com/cn/articles/webkit-for-developers>

## 机制

通过信号和槽机制让js和C++交互。

## 核心类

QWebview、QWebpage，QWebframe。

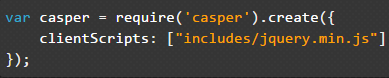
# 提速相关

## Phantomjs中关闭图片加载

–load-images=false

## jQuery引用

在创建casper实例时可以配置引入jQuery使得每个page都可以使用：



注意：在注入脚本时不能使用http协议，应该是用文件系统路径，一般使用本地文件系统。

# 关于casperjs中的repeat接口

## 流程

reapeat实际的操作是将调用指定次数的then接口。then接口的实现依赖事件来完成。casperjs.js文件中有对事件模块的启用。

var events = require('events');

# Node

## Node简介

刚才说到Node，这个和javascript的关系是，Node是一个运行环境，这个环境自然有javascript的解析引擎。所以Node 就是一套能运行js代码的环境。

## util模块

util模块用来帮组继承EventEmitter。

util.inherits(Radio, EventEmitter);

Radio是一个构造函数。可以继承EventEmiiter构造函数。

## 事件

Events模块是Node实现的，是对publish/subscribe模式的一种实现。是对象之间通信的一种模式。毕竟对象在js这类基于对象的语言中很重要。Event模块的实现中，其中EventEmitter是一个对象好像很重要。对象通过emit函数触发，在触发前，对象得先调用on函数绑定事件和回调函数。同一个事件，默认允许最多的绑定回调函数个数是10个。

## 继承

一般对象并不直接使用EventEmitter，而是通过继承EventEmitter对象的原型。长这样自定义的对象就能调用on和emit接口。

## JS的几个特点

js没有类的概念，类似的通过原型来完成，js里函数也是对象。