

演講

這是來自 GitHub 專案 <https://github.com/lzwjava/Keynotes> 的 README.md 。

Keynotes：

- Git 高級操作和原理
- WebSocket
- UnitTest
- 毫無保留的直播技術

歡迎交流，有問題請提 issue，看到後將盡快解答。

Live

毫無保留的直播技術

在 Live 目錄下。

Git

2016.5.3 斗魚直播

WebSocket

2016.4.23 技術交流會分享。

62 頁

UnitTest

2015.12.20 在斗魚 TV 上直播的 Keynote，關於單元測試、自動化、好用工具等。視頻: <http://reviewcode.cn/video.html?vid=1>

40 頁

...

Git 高级操作和原理

李智维

Figure 1: qq20160503-0 2x

Git Objects

```
$ echo 'test content' | git hash-object -w --stdin  
d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

```
$ find .git/objects -type f  
.git/objects/d6/70460b4b4aece5915caf5c68d12f560a9fe3e4
```

```
$ git cat-file -p d670460b4b4aece5915caf5c68d12f560a9fe3e4  
test content
```

- hash-object, 把数据保存到 .git 目录的命令
- -w, 写入对象, 否则只是返回 key
- –stdin, 从标准输入中读取
- d670..., 40个字符的 checksum
- cat-file, 查看 Git Object 的瑞士军刀

Figure 2: qq20160502-0 2x

```
$ irb
> content = "what is up, doc?"
=> "what is up, doc?"

> header = "blob #{content.length}\0"
=> "blob 16\u0000"

> store = header + content
=> "blob 16\u0000what is up, doc?"

> require 'digest/sha1'
=> true

> sha1 = Digest::SHA1.hexdigest(store)
=> "bd9dbf5aae1a3862dd1526723246b20206e5fc37"

> require 'zlib'
=> true

> zlib_content = Zlib::Deflate.deflate(store)
=> "\x9CK\xCA\xC90R04c(\xC9F,0\xC8,V(-\xD0QH\xC90\xB6\ax\x00_
\x1C\ax\x9D"

> path = '.git/objects/' + sha1[0,2] + '/' + sha1[2,38]
=> ".git/objects/bd/9dbf5aae1a3862dd1526723246b20206e5fc37"

> FileUtils.mkdir_p(File.dirname(path))
=> [".git/objects/bd"]

> File.open(path, 'w') {|f| f.write zlib_content }
=> 32

$ git cat-file -p bd9dbf5aae1a3862dd1526723246b20206e5fc37
what is up, doc?
```

用 ruby 演示 Git 对象的存储

- header 和具体内容一起构成最后的保存对象
- SHA1 得到 40 个字符, 前 2 个作为子目录, 后 38 个作为文件名
- zlib 压缩
- cat-file 检验是否保存成功
- blob 的内容可以为任意内容, 但 commit 和 tree 格式要求严格

Figure 3: qq20160502-3 2x

内部对象

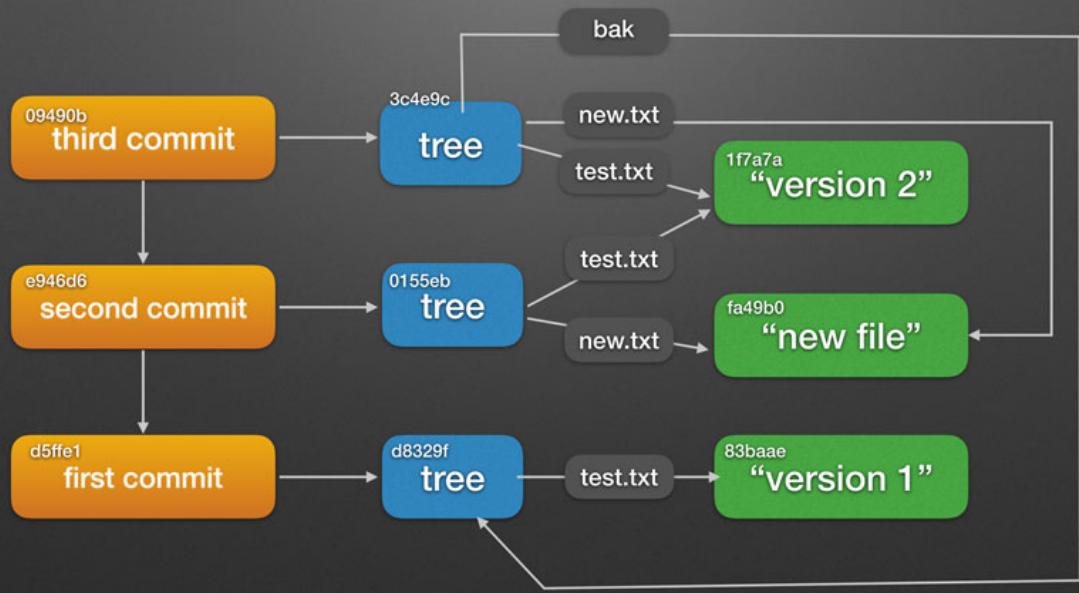


Figure 4: qq20160502-2 2x

讲讲 WebSocket

李智维
2016.4.23

Figure 5: qq20160423-1 2x

Credit

感謝彥祖、ufosky、tang3w、sunng87、iOS 程式員、冉神。感謝 LeanCloud CTO 允許我講內部測試流程。

内容

- WebSocket 当今的应用
- WebSocket 的历史由来
- iOS 平台如何使用 WebSocket
- 详解 WebSocket 协议
- iOS 平台如何实现 WebSocket 协议

Figure 6: qq20160423-2 2x

SRWebSocket

```
62 @interface SRWebSocket : NSObject <NSStreamDelegate>
63
64 - (id)initWithURL:(NSURL *)url;
65
66 // SRWebSockets are intended for one-time-use only. Open should be called once
67 // and only once.
67 - (void)open;
68
69 - (void)close;
70 - (void)closeWithCode:(NSInteger)code reason:(NSString *)reason;
71
72 // Send a UTF8 String or Data.
73 - (void)send:(id)data;
74
75 // Send Data (can be nil) in a ping message.
76 - (void)sendPing:(NSData *)data;
77
78 @end
```

Figure 7: qq20160423-6 2x

帧协议

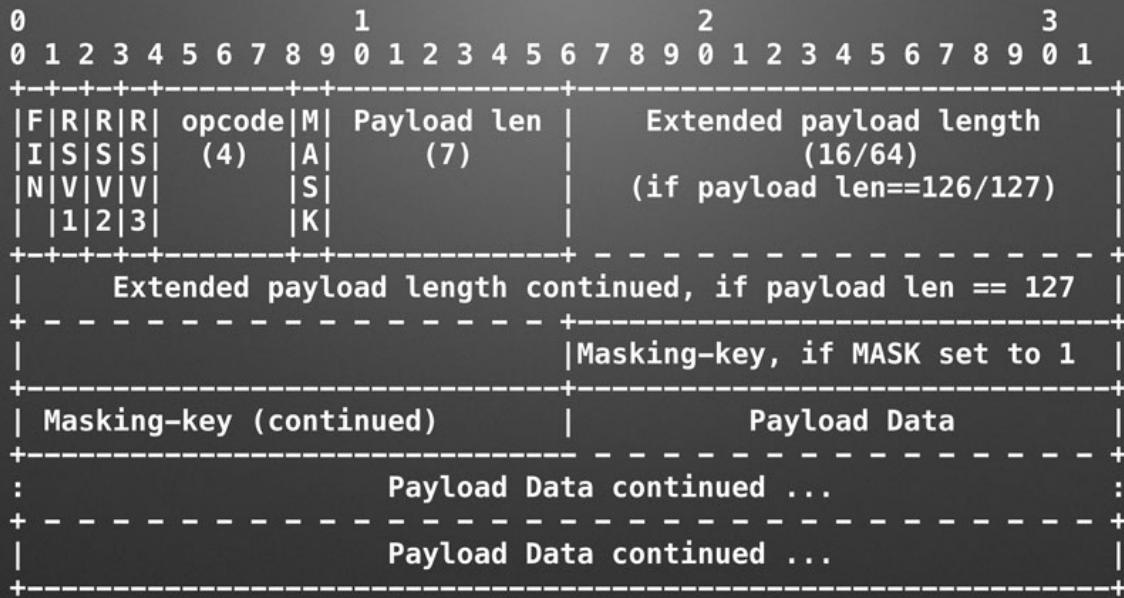


Figure 8: qq20160423-4 2x

帧协议

- Mask: 1位。如果值为 1, masking key 会出现在 masking-key 中。所有客户端往服务器发送的帧都必须设为1
- Payload length: 7位或(7+16)位或(7+64)位, 1)0-125, payload length
2)126, 则 16位整数代表的值为 length 2)127, 则64位整数为length
- Masking-key: 0 或 4 字节(32位)。如果 Mask 为 1, 则存在, 否则不存在。
- Payload data: (x+y)bytes, Extension Data + Application Data
- Extension data: x bytes, 必须满足握手阶段协商好的长度
- Application data: y bytes

Figure 9: qq20160423-5 2x

例子(1)

- A single-frame unmasked text message
0x81 0x05 0x48 0x65 0x6c 0x6c 0x6f (contains “Hello”)
- 0x 表示16进制
- 0x81, 也即 1000 0001, 对应表来分析。1 即 FIN, 表示这是一个完整的帧。0001 表示 Opcode, 为1, 表示 Text Frame, 这是一条文本消息
- 0x05, 即 0000 1001, 表示 5 , 也即长度为 5
- 0x48 对应 ASCII H, 0x65 照推
- 所以上面的数据意思是, 一条完整消息, 是文本, 长度为 5, 具体内容为 Hello

Figure 10: qq20160424-0 2x

Opcode

- %x0 : continuation frame
- %x1: text frame
- %x2: binary frame
- %x3-7: 保留给未来的非控制帧
- %x8: connection close
- %x9: ping
- %x10: pong
- %xB-F: 保留给未来的控制帧

```
51
52 typedef enum {
53     SR0pCodeTextFrame = 0x1,
54     SR0pCodeBinaryFrame = 0x2,
55     // 3-7 reserved.
56     SR0pCodeConnectionClose = 0x8,
57     SR0pCodePing = 0x9,
58     SR0pCodePong = 0xA,
59     // B-F reserved.
60 } SR0pCode;
```

Figure 11: qq20160423-8 2x

长轮询

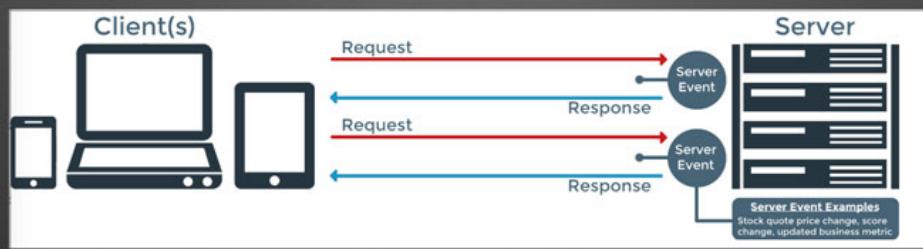


Figure 12: qq20160423-3 2x

资料

- WebSocket RFC : <https://tools.ietf.org/html/rfc6455>
- 知乎《WebSocket 是什么原理?》: <https://www.zhihu.com/question/20215561>
- SocketRocket: <https://github.com/square/SocketRocket>

Figure 13: qq20160423-9 2x

覆盖率

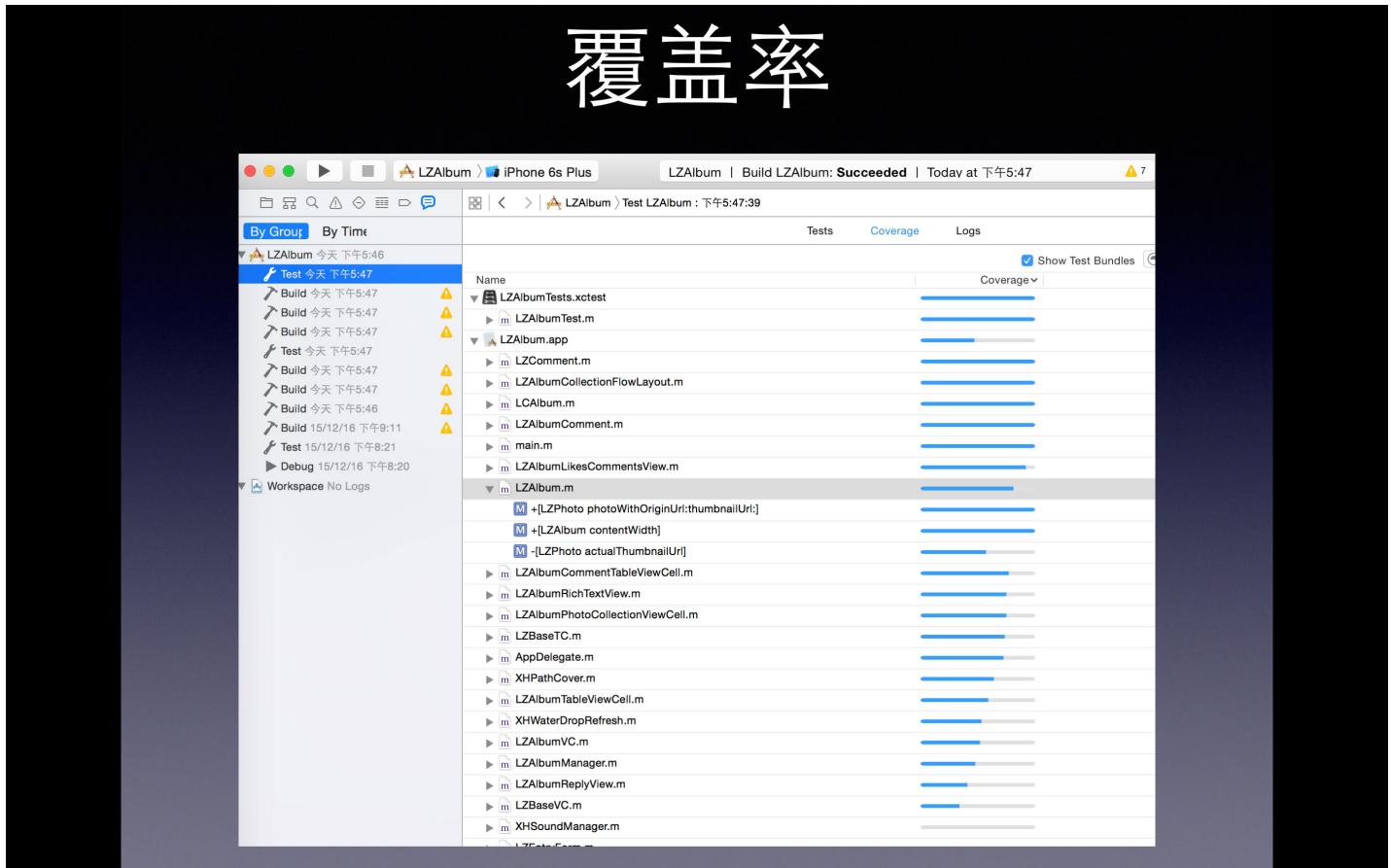


Figure 14: key1

输入暗号触发测试

The screenshot shows a GitHub pull request titled "add auto test script #428" which has been closed. The pull request details include:

- Conversation: 6 messages
- Commits: 1
- Files changed: 1

The pull request history shows two comments from the user "lzwjava":

- Comment 1: "No description provided."
- Comment 2: "lbuild-me" (with a timestamp of 88d2100)

Below the pull request, there is a configuration section for "Use github hooks for build triggering". The configuration includes:

- A checked checkbox labeled "Use github hooks for build triggering".
- A "Trigger phrase" input field containing the value "\Q!build-me\E".
- Two unchecked checkboxes:
 - "Only use trigger phrase for build triggering"
 - "Close failed pull request automatically?"

Figure 15: key2

零碎的知识

- lipo 使用
- 清空所有的生成文件，Clean Build Folder
- Xcode 快捷键，根据当前文件展开左侧导航、Open Quickly、查看 Macro 预编译、.h 与 .m 文件跳转
- Pod 高级用法
- 如何制作 Framework
- Xcode Configurations
- Reveal In GitHub 插件
- Instrument 工具使用，定位代码

Figure 16: key4

如何写单元测试

- 模块化代码，数据层和 UI 层分离
- 最少的测试代码达到最高的覆盖率
- 异步处理
- 框架选择
- 覆盖率

Figure 17: key5