mikutterの薄い本 vol.6.0

特報！！

ておくれの原点が

たくさんの「ておくれ」を連れて帰ってきた！

teokure

ポケット・ミク接続で
みくったーちゃんがしゃべりだす！！

指導教官が Twitter を始めたときの危機管理

まえがきにかえて
@brsywe

1. クライシス

ある冬の日の出来事である。RSS リーダーで読んでいた指導教官のブログに驚くことが書かれていた。

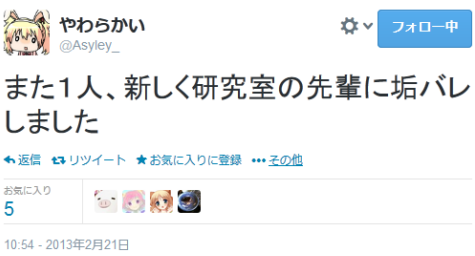
Twitter を始めました

以前、飲み会の席で指導教官に「Twitter ってどうなの」と訊かれた際、わたくしは「大学教授は一般的に実名で発信しますし、いわば叩かれることにも慣れている筈ですから Facebook のほうがおすすめですよ。Twitter は実名でやるのに向きません」と答えたのである。Twitter で日常的に（平均的なユーザーに比し）Tweet 数も多く、いくらかの信念をもって Twitter で遊んできたわたくしとして指導教官に Twitter を始められるのはいくらか危険であった。

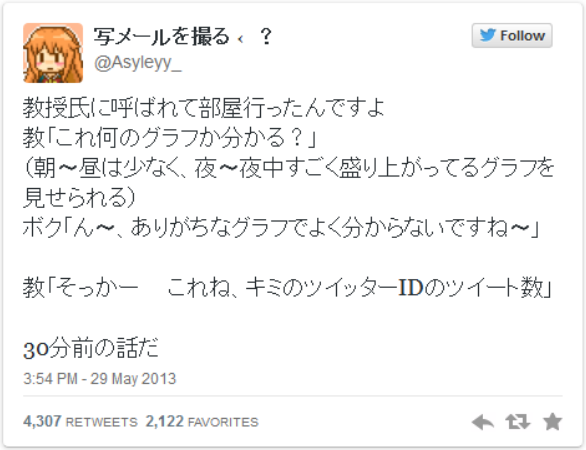
然し事態は防げなかった。指導教官が Twitter を始めてしまった事実は変えられない。さて、今後どうすべきか。それを考えるために本稿を執筆することとする。

2. 先行研究

身近なところによい題材があった。教授にバレた人がいる。@Asyley_ （以下、あしゅりーおじさん）である。



↑ラボの人に新たにバレた Tweet。



https://twitter.com/Asyley_/status/304408813803540480

↑垢バレ後の悲しい Tweet。

https://twitter.com/Asyleyy_/status/339635860704206849

あしゅりーおじさんの垢バレは、研究室の人にバレて、それが教授に垂れこまれて教授バレという経緯を辿った、とのことである。

教授に垢バレしない方法について、あしゅりー氏は「*研究関連の事をつぶやかない　ラボメンに垂れ込まれないためにアカウントの存在を隠す*」ことだと述べている[1]。

炎上によるアカウントの「拡散」、教授の検索能力どうこうよりもっと身近なところから垢バレするものであるということを改めて思い知らされる。あなたはどうだろうか、バレたくない相手とつながっている follower はいないだろうか。以下でバレないための対策について検討を加えることとする。

1 https://twitter.com/Asyley_/status/433186992986144768

3. バレないための方策

あしゅりーおじさんの垢バレ事案からバレないための方策を検討する。

- 凍結に追い込む

Twitter を始めて直後は、一部のアカウントを除き follower が少ない。これに乗じて複数アカウントで対象となるアカウントをブロックするという方法が考えられる。

しかし、なんの交流もないアカウントに対して事前に、しかも複数アカウントからブロックするのは「通常」ならば心理的な抵抗があるうえ、複数アカウントで同じ行動を起こすというのもスパム行為に近いものがある。更に、数アカウントからブロックする程度では実効性がなく、通常人には為しえない方法だ。

- フォロイーすべてをブロック

対象となるアカウントがフォローしているアカウント（フォロイー）をすべてブロックすることによって対象に自分の Tweet が RT によって流れることを防止するという方法が考えられる。

数十アカウント程度であればこの方法も「アリ」だが、フォロイーは当然に増えていくと考えられるため、そのたびにブロックする数を増やさねばならぬこととなる。無論、そのうちに自らのフォロワーを follow することも考えられ、ギスギスしてしまうことがあるだろう。

- 他大学の学生になりしまし

今回、事例が発生したときに偶然にも取っていた行動でもあり、発生後にも更に嘘を重ねた行動でもある。

一例として、Twitter-er が多い TKB 大学の学生になりすますというのはどうだろうか。エア TKB として活動しているわたくしの場合には、TKB 大学周辺の飯屋の Tweet を時折 RT し、bio には実在する学類の略称を表記し、TKB の学生を複数釣っている。それにより、たまたまなんらかの形で指導教官にっ見つかったとしても、まさか自分の大学の、更には教え子である、ということに気付かせないものである。わたくしであれば「coins」: 情報科学類を装っている。

TKB の Twitter-er を見ればわかるとおり、なりすます学類の関連のことを Tweet する必要性は乏しい。

もっとも、指導教官が TKB 大学の略称を知っているとも思われないのでプロフィールに（虚偽の）大学名を書き、Location に虚偽の、または架空の地名を書くことが望ましいと云える。

4. この検討の転用

この方法は指導教官にバレない偽装についてだが、勤め先や Twitter のアカウントを知られたくない知人に対しても威力を発揮する。

5. まとめ

炎上に比べて比較的頻度が高い「垢バレ」。みなさんはどのような対策をしているだろうか。本稿をきっかけに何かしら良案が浮かべば @brsywe までご連絡願いたい。

↑vol.5PDF 版のときのロゴ

さてこの本がでているときに上のロゴのシャツはでているのだろうか。さて今号もどうにか発行にこぎつけた『はぴねす！えもーしょん』楽しんでいっておくれ。

Inside of 田端でバタバタ

@misodengaku

まえがき

ほとんどの方ははじめまして。田端でバタバタ bot (@Batabata_Tabata) を制作、運営している @misodengaku です。2013 年末だったかその辺りに薄い本別冊で何か書かないかと室長に誘われどうしようかなどと言っていたらジョークグッズによる圧力をかけられたので書かせていただくことになりました。

この記事では田端でバタバタ bot (以下田端 bot) のこれまでの歩みと技術的観点から見た田端について書いていこうと思います。

田端のはじまり

2013 年 4 月から私が山手線を使うようになり、毎日田端を通過するような生活が始まりました。やることもないので当然電車内ではクソみたいなツイートを発射しまくっていたわけですが、そんな中で私が思いついたツイートが「田端でバタバタ」でした。

田端のあゆみ

公式な記録なんか当然残っていないので当時の資料を掘り返していく形になるので完全ではありませんが、田端 bot の周りで起きた出来事を時系列順にまとめていきます。

2013/04/xx

私が山手線を使うようになる。おそらくこの頃「田端でバタバタ」を思いつく。

2013/07/08

今まで何回バタバタしてんだろ、と気になり回数をカウントする bot を制作。

当時は身内でゲラゲラ笑いながら遊んでいただけなので回数規制などは一切なく、誰でもツイートさえすれば連続でもなんでもカウントされる仕様で、screen name は@Tabata_Batabata だった。

2013/07/09

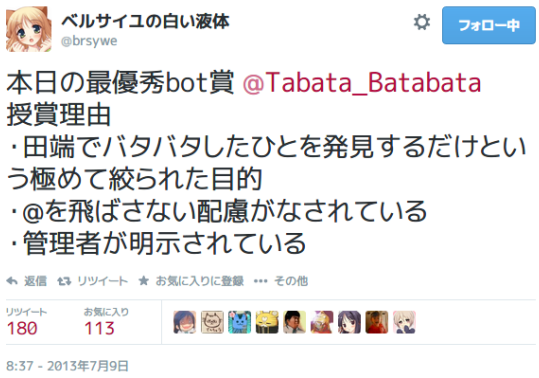
室長（@brsywe）が田端 bot の宣伝ツイートをする。

RT 数等からわかる通りこれが原因で意味不明な人気を博す。

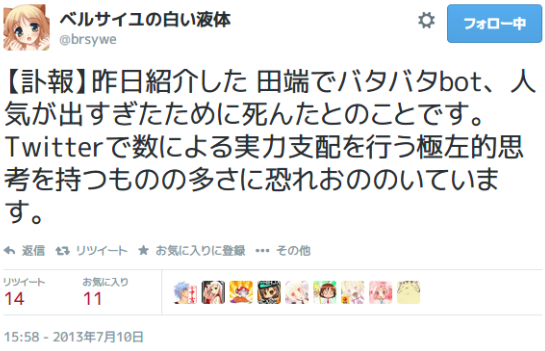
（参考資料：http://goo.gl/hB992L）

2013/07/10

前日のツイートが原因で田端 bot の存在が一般に知られユーザー数が激増、そのせいか @Tabata_Batabata が凍結される。急遽@Batabata_Tabata というアカウントを新設し、環境の移行と再発防止策の検討を始めた。



資料 1. 室長の宣伝ツイート



資料 2. 凍結後の室長のツイート

2013/07/11 ~ ?

bot 全体としてのツイート数リミッターや連続でのバタバタを無視する仕様などを追加。

2013/08/17

@DigitalCuko が「田端でバタバタ」と発言し身内で騒ぎになる。

資料 3. バタバタするキュー子

~現在

特段面白いことも起きず、運用も安定したため日々の保守が繰り返される。

2014/05/29 2:22 現在で bot が捕捉したバタバタの総回数は 474589 回、データベースに保持されているアカウント数は 47817 にまで成長した。

田端のしくみ

田端 bot は mikutter でもおなじみの Ruby を使って書かれています。bot の運用環境としては、Ruby 2.1.1p76（執筆時点）を Hyper-V で仮想化した Gentoo Linux 上で動作させています。

いちおう bot 本体のソースコードは公開していますが、そもそも田端 bot は私が Ruby の学習用に書いたものであり、現在のソースは学習過程で増改築を繰り返しまくってハチャメチャなことになっているのでここには載せません。

もっと突っ込んだ内容としては、田端 bot では tweetstream Gem を利用して「田端でバタバタ」というワードを条件として設定した filter stream に接続します。ここで Twitter API のクソさが爆発し、マルチバイト文字がデリミタとして認識されないという仕様がありますが、ちょっとこれは私にはどうしようもないのでユーザーには「仕様」として我慢してもらうことにしました。Bot 自体は daemon-spawn Gem で daemon 化していますが、Linux 本体を再起動させたときに手動で立ち上げなければならないので改善を検討中です。DB にはなんとなく sqlite3 を選びましたが、まぁバックアップも楽だしいいんじゃねという感じで正解だったかなと思っています。

田端の規制

田端の規制ポリシーは公開されている bot のソースコードを読み解くことで理解できます。

具体的には、前回のバタバタから 120 秒以上経過している場合はアカウントごとに管理される回数に+1 され、さらに田端 bot が前回ツイート（「takedashi さんが xx 回バタバタしました。」という感じのツイート）してから

85 + (0～9 の乱数)秒経過していた場合は田端 bot により回数がツイートされます。

120 秒の根拠としては、田端から西日暮里か駒込までの乗車時間が乗換検索によれば 1 分とのことなので、往復した場合を考えてその倍の 2 分、120 秒に設定しました。85 秒に関しては、規制されないギリギリのラインを考えて設定した値ですが、そもそも規制の条件自体が不明確なので

これが最適な値かどうかはわかりません。乱数は bot 弾きのため導入しました。

また、自動投稿に特化したクライアントや爆撃機能を持つクライアント等は NG クライアント、爆撃等の bot に対する攻撃を行ったアカウントは NG ユーザーとして登録され、どちらかから投稿されたツイートに関しては完全に無視されます。

なぜ規制するのか

一度バタバタしたら一定時間は反応しない仕様など連続投稿を嫌う理由は、田端のあゆみでも触れた通り田端 bot が一度凍結された過去を持っているからです。いつでも誰でもバタバタを記録できるように、あとは単純にめんどくさい等の理由で回数に縛りをかけ、Twitter 側のクソ運営に目を付けられないように配慮しています。

また、田端でバタバタのあそびかた (http://totori.dip.jp/tabata/faq/) 等でも明言している通り自動投稿や bot による回数稼ぎなども禁止しています。これは単純に bot のポリシーである「回数を記録する」に反しているからであり、クソみたいな bot が無駄に発したバタバタで私達のような実際に田端駅からツイートを発している人々のバタバタがツイートされる機会が失われてはならないからです。あとは単純に気にくわないからです。やめろ。

Twitter for iPhone 等のわかりづらいコンシューマーキーを利用して bot を運用してもだいたいこちら側からすればわかっています。繰り返しになりますが面倒なのでやめろ。

そもそも田端でバタバタとは

Google で「田端でバタバタ」と検索すると「田端でバタバタ 意味」や「田端でバタバタ 元ネタ」がサジェストされます。あえて明言してきませんでしたが**田端でバタバタという言葉の意味はありません。**某知恵袋で解答されている ”意味”は正しい正しくない以前に面白くありません。その辺りを自由にやってもらうために明言してこなかったのですが、やはり人が増えるとダメですね。

あと元ネタに関してですがこれに関しては非常に微妙です。「田端でバタバタ」という言葉自体は当然誰でも思いつくようなただのダジャレですし、実際 Google などで検索してみてもかなり前に発言している人が見受けられます。私自身は何かを見てツイートしたとかではなく単純に思いついてツイートしたので私が考えたと言えなくはないですが少し無理があるでしょう。田端でバタバタというワードはかなり昔に自然発生していたが、それを変な形で流行らせたのは田端 bot が原因あたりの認識が妥当だと思われます。

田端の今とこれから

よくわからない流行のおかげで田端 bot では毎日だいたい 1500～1900 回のバタバタを記録する変な人気を得ました。現実的に考えてもうこれ以上は伸びないでしょうし伸びられても困ります。そもそも 1 年近く数字が安定しているのが未だに理解できません。

現在 bot は Ruby で書かれており、非常に安定して稼働を続けていますが、余りにもコードが汚くまともに読めないような状況です。書き直そうかとも思っていますが安定して動いている以上別にいいかなーという感じです。

なにか面白いことを思いつけば機能追加するかもしれませんが正直私自身飽きている状態なのでやらないと思います。

さいごに

mikutter にほとんどというか一切関係ない記事でしたがいかがでしたでしょうか。これを機会に田端でバタバタ bot を知ってもらえれば嬉しいですが特に何も得をしないどころか運用コストだけがかかっている状態なので投げ銭していただけると助かります。

お読みいただきありがとうございました。

連絡先

Twitter: @misodengaku　Mail: misodengaku@gmail.com

投げ銭 (ウィッシュリスト): http://goo.gl/h0rV5k　→

嫁タブプラグインの解説

あっきぃ（@Akkiesoft）

本稿では、嫁タブの特徴や、その実装について解説します。と言いたいところでしたが、ソースを見返したらショボすぎて「こりゃあ、おこがましいな」と感じたので、概要と一つだけ語っておきたい点について語る程度にとどめさせてください。

嫁タブの概要

嫁タブプラグイン（以下、嫁タブ）は、mikutter 上に任意の画像を表示するだけのプラグインです(図 1)。いわゆる「俺の嫁」などを表示して Twitter のタイムラインに添えることで、精神衛生を浄化し、より快適な mikutter ライフを送ることを目的としています。嫁タブは、github からダウンロードができます。

https://github.com/Akkiesoft/mikutter_yometab

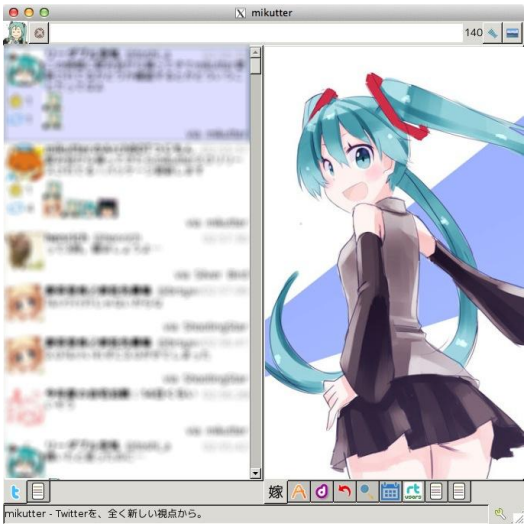
嫁タブの特徴「可変タブ」

本プラグインの特徴は特になく、mikutter らしいておくれたプラグインの一つと自己評価しています。

mikutter のプラグイン的に珍しいであろう点がひとつを一つ挙げるとすると、タブのアイコンが設定で変化すると言う点があります。

嫁タブはデフォルトでは、「嫁」と書かれたタブに嫁の画像を表示します。しかし「嫁よりも妻派」「嫁がいない」「妹派」「おかの志向」などの多様な需要に応えるべく、本プラグインでは設定を行うことでタブ名を変更できるようにしました。これにより、「妹の画像を表示したいのに嫁と言う名前のタブに表示される」と言った悲劇を回避することが可能になります。

図 1　嫁タブプラグイン



可変タブの実装

設定に応じてタブ名を変化させることは、特別トリッキーなことをしているわけではありません。実装はコード 1 のようになります。

このような使い方は、既存のプラグインではそれほど例がなかったと記憶していますが、他にも応用を利かせられる可能性があると考えられます。

例えば、天気予報プラグインなどはどうでしょうか。タブの中では詳細な情報が得られるが、タブを開いていない時でも現在の予報や気温など情報をアイコンとして表示できる……と言ったように、タブが開かれていなくても最低限の情報が得られるタイプのプラグインづくりに役立てられるのではないでしょうか。

ただし、軽く検証を行ってみた結果、実装は少し難しくなるかもしれないという印象を受けました。

TODO

ドヤ顔で語っておいてなんですが、嫁タブは適当に各所のソースコードを参考にして作られたため、不完全な部分が多々あります。特に、設定変更後は mikutter の再起動が必要な点はなかなか致命的です。今後改善していけたらと考えることはありますが、面倒なのでたぶんやりません。超いい感じのプルリクエストなどは歓迎ですのでお待ちしております。

コード 1　タブの実装部分の抜粋

```
yometab_table = {
        'yome'   => '嫁',
        'tsuma'  => '妻',
        'musume' => '娘',
        'imouto' => '妹',
        'kanojo' => '彼女',
        'jis'    => '㋮'
}
yometab_name = UserConfig[:yometab_name]
if yometab_name == '' then yometab_name = "yome" end
tab(:mikutter_yometab, "#{yometab_table[yometab_name]}タブ") do
  icon = File.expand_path(File.join(File.dirname(__FILE__), "#{yometab_name}.png"))
  set_icon icon
  # (中略)
end
```

mikutterはWindowsでも動きます！マジで。

もぐの（@moguno）

1. はじめに

ちす！もぐのと申します！！

僕とmikutterさんとの出会いは2011年に開催されたOSC Kobeに遊びに行ったところから始まります。
「世の中には技術とイチャイチャしてる人たちが一杯いるんだ！すげー！」と、非常にカルチャーショックを受けました。

特につついさん(@tsutsuii)の「海外からATARI用の自作Ethernetカードが送られてきたのでNetBSDのドライバを書いた」話がエキサイティングでした。（http://www.ceres.dti.ne.jp/tsutsui/osc2011kobe/NetBSD-EtherNEC.html）C言語コードで笑いを取る人って初めて見ました。

その後、OSCについて言及している方々を色々フォローしている内に、mikutterなるものの存在を知り、

もぐの@いっそウサギになりてー
@moguno

多分これは下名が辿り着きたかった理想郷。でも言ってること半分も分からんから取り敢えず使ってみる！ / mikutter http://htn.to/Ek3LCe

うわ、なんか理想郷見つけちゃてるよこの子…。

それからちょこちょことプラグインを書いている内にRubyの黒魔術に出会い、

「くくく…モンキーパッチか。これさえあればmikutterプラグインで世界を掌握する事すら可能っ！」

と、頭の中の何かがSEGVしてしまい現在に至ります。
こうやって見るとかなり奇麗にておくれていますね。

さて、今回はておくれ界で最もマイナーなOSでmikutterを動かした話をさせてもらおうと思います。

2. Windows でも mikutter したい！

きっかけは2014年1月に発売された低価格Windows 8.1タブレット、DELLのVenue 8 Proの購入でした。出張中の動画鑑賞をメインに、マルチタスクでTwitterとかもやりたいなと。そこでWindows用のTwitterクライアントを物色したのですがどうもしっくり来る物が無い。どうせだったら使い慣れたmikutterが動いたらいいのにな。

と言う訳で、TLでちらほら見かけていた**「mikutterがWindowsでサクサク動いた」**的な情報を検証してみる事にしました。

そしてWebを彷徨うこと数時間、最近の成功事例と思われる一稀さん(@kazuki_kaihatu)のブログに辿り着きました。ブログにはRubyインタプリタの選定から日本語でツイートするまでの流れが詳しく纏まっており、私は都市伝説が真実であると確信しました。

実際、手順通りにセットアップを行うとmikutterの見慣れたタイムラインが現れました。起動成功です。しかし、何分もしないうちにウインドウが「（応答無し）」になって固まってしまいました。

ううむ。うちの環境では何かがおかしいようです。

動かないウインドウを前に途方に暮れていると、コマンドプロンプトにログメッセージがポロポロ・・・あ、これGTKがいるメインスレッドが固まってるだけじゃね？

「…もしかするとこれ直せるのでは？」

気が付くと、私は色んな意味で味わい深いmikutterのソースの海に飛び込んでいました。

結果的にこのダイビングは大成功し、Rubyコードの変更のみでmikutterを安定動作させる事が出来ました。
（Windows 7上で5日間連続稼働を確認しています。）

「よし、じゃあこれプラグインにしちゃおう。」

こうして、mikutter-windowsは誕生したのでした。

3. mikutter-windows について

3.1. なにこれ？

mikutter-windowsは、Windowsでmikutterを動作させるためのパッチ集＆インストーラです。

パッチ集と言ってもプラグインとして実装しているため、mikutter本体のソースコードを変更する事無く導入が可能です。「自分を動作させるためのプラグイン」とか、なんともけったいな物を提供出来ちゃうmikutterのポテンシャルは本当に半端ないと思います。

また、mikutterの動作環境を整えるにはドス黒いコマンドプロンプトでの作業が必須なのですが、mikutter-windowsでは必要な手順をインストーラ(setup.rb)にまとめることで、お手軽簡単にmikutterがインストール出来る様になっています。

3.2. 動作環境

mikutter-windowsプラグインは下記環境での動作を確認しています。

(1)Windows
・・Windows 8.1 Pro (64bit)、 Windows 8.1 (32bit)
・・Windows 7 Pro (64bit)

GTKがタッチパネルでのスクロールに対応していないので、Win8.1タブレットではちょっと操作性が悪いかもしれません。

(2)Ruby
RubyInstaller for Windows Ruby 2.0.0 (32bit)

64bit版はGTKの導入が手間なので、32bit版を使いましょう。

(3)mikutter
0.2.2以降

3.3. mikutter-windowsのインストール

インストールの詳細はmikutter wikiの「Windowでのmikutterインストール」を参照して下さい。(http://yuzuki.hachune.net/wiki/MikutterInstallBattleForWindows)

zipで配布されているフリーソフトをインストールした事が有る方なら、多分OKです。

4. mikutter-windows ってなにやってるの？

4.1 Windows特有の問題へのパッチ

mikutter-windowsではWindows特有の各種問題をmikutterのソースコードを動的に変更することで対処しています。巷で噂のモンキーパッチと言う奴です。

(1)起動後、数分でフリーズする問題の回避

前述の通り、Windowsでmikutterを起動するとだいたい数分でメインスレッドが固まり、ウインドウが「(応答無し)」になる現象が起こっていました。

調査のために下記のコードで定期的にメインスレッドの状況を表示したところ、Ruby標準クラスのPStoreがFile::flock()メソッドに入ったままひたすら返って来ない状態に陥っていることが判明しました。

```
Thread.start {
  loop {
    puts Thread.main.backtrace
    sleep 10
  }
}
```

「あー、多分誰かがファイルの排他ロックを持ったまま離さない的なありがちな不具合だろうなー」と色々調べてみるも原因不明。

恐らくmikutterよりももっと強大な敵(Ruby本体かMinGW辺り？)が行く手を阻んでいるようです。困りました。

あれやこれやと悩むこと数日。mikutter-windowsはこの問題に対する見事な解を見いだしました。

```
class File
  alias :flock_org :flock

  # flockがロックかかってないのに戻ってこなくなるので、そんなメソッド無かったことにする。
  def flock(ope)
    return 0
  end
end
```

言う事聞かないメソッドは華麗にスルー！
戦うだけが正義じゃない！

…一応、PStoreの中にも排他処理があるFile::flock()が無くても大丈夫だろうと言う判断しています。

でも、プロセスを跨いだロックは掛からないのでmikutterを複数起動するとこけます。いい子はTwitterクライアントを複数起動しちゃうようなておくれた事はしないよね？

なお、2014年6月1日にリリースされ**ると誰もが思っていた**mikutter 3.0では設定の保存にPStoreを使わなくなったので、この問題は発生しません。しかしながら、サードパーティのプラグインでPStoreを使っている物があるらしいので、これはそのままにしておこうと思います。

4.2 Windowsと言う「特殊な」OSへの配慮

LinuxやNetBSDに代表されるUNIX(っぽいOS達)と、世界一普及しちゃってるデスクトップOSのWindowsはその出自を完全に異にしており、大小色々な差異が存在します。そのため、UNIX（っぽい）向けのソフトウェアであるmikutterをWindowsで動かすためには、その差異を埋める努力が必要でした。

(1)文字コード
mikutterのソースコードは、UNIX（っぽい）環境でのデファクトスタンダードである「UTF-8」と言う文字コードを前提に記述されています。一方、WindowsではMS-DOSから脈々と続く「Windows-31J（Shift-JISの亜種）」を採用しており、両者の文字コードにミスマッチが生じています。

mikutterの動作環境であるRubyでは、異なる文字コードを混ぜて使用すると例外が発生してプログラムが終了してしまうので、文字コードのミスマッチは深刻な問題となります。

例えば、一稀さんのブログで指摘されている日本語のメッセージを投稿すると落ちる、いわゆる「141行目」問題は「UTF-8」な投稿メッセージを「Windows-31J」のコマンドプロンプトに表示しようとしたせいで生じます。

```
inspected result must be ASCII only or use the same encoding with default external
```

さらにタチの悪いことに、文字列が半角英数だけで構成されていた場合、文字コードが「ASCII-8BIT」と判定される場合が有ります。「ASCII-8BIT」は「Windows-31J」と互換性があると見なされるため、混在しても例外は発生せず普通に動作してしまいます。

従って、先の141行目問題も投稿メッセージが「teokure」とかだと問題なく動いてしまうので、問題の存在を見逃しがちになってしまうんですね。泣きそうです。

mikutter-windowsでは文字コードのミスマッチがありそうなメソッドに都度モンキーパッチを当てて辻褄を合わせています。目に付く問題は取り除いていますが、日本語を含むパスにmikutterをインストールした場合にアウトなパターンが有ります。もうちょい退屈になったら対応しようと思っています。

ちなみに、例の141行目問題はmikutter 3.0で対応済みです。

(2)改行コード
もう一つ、UNIX（っぽい）とWindowsとの大きな違いにテキストファイルの改行コードが有ります。UNIX（っぽい）では”LF”1バイトで改行を表現しますが、Windowsでは”CR + LF”の2バイトになります。

この問題はさっきの文字コードよりも伝統的かつシンプルであり、プログラミング言語側で面倒を見てくれるのが一般的です。

例えばRubyではFile::open()に改行コードの自動変換機構があり、ファイルを読むときにはOSの改行コードをLFに、逆にファイルに書く時にはLFをOSの改行コードに変換してくれます。この仕組みにより、プログラマはOSの改行コードを気にする事なく、LF前提でソースコードを書けばOKとなっています。

この機構の注意点は、画像ファイル等のバイナリファイルを開く時に改行コードの自動変換が働いてしまうと、ファイル中のCR(0x10)とLF(0x13)がLF(0x13)に変換されてしまい、バイナリファイルが壊れてしまうことです。これを回避するには、バイナリファイルを開くときはFile.open()に改行コードを変換しないようにオプションを指定する必要が有ります。

この挙動はCR + LFなWindows界隈では有名なのですが、OSの改行コードがLFなUNIX（っぽい）ではあまり問題にならず、バイナリファイルに改行コード自動変換を効かせて開いている事が結構有ります。

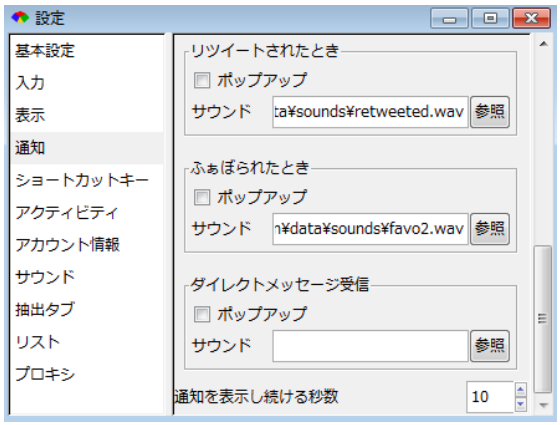
mikutterについては、画像キャッシュファイルの読み書きに改行コードの変換が掛かっていたため、高頻度で画像がError表示になると問題に繋がっていました。

mikutter-windowsではFile::open()を使っているメソッドにモンキーパッチを当てて対応していましたが、これについてもmikutter 3.0でmergeされました。
0.2のサポートを切る際に削除しようと思います。

4.3 サウンド

mikutter-windowsはWindowsで通知音を鳴らすためのサウンドプラグインを内蔵しています。これによりWindowsでもみくったーちゃんの可愛い「ふぁぼ♪」を聞くことが出来ますね。

イベント発生時の通知音は設定の「通知」カテゴリで(mikutterのインストールフォルダ)/core/skin/data/sound/以下のwavファイルを設定して下さい。



実装的にはWin32APIモジュールを使って、古のWindows APIであるPlaySound()を呼び出してサウンドを鳴らしています。

4.4 インストーラ

mikutter-windowsのインストーラでは次の処理を行っています。

(1)必要なライブラリの自動インストール
bundlerのインストールとmikutterのGemfileの処理を行って、必要なgemをダウンロード＆インストールします。

従って、Windowsユーザは、**「gem？何それ新手の妖怪人間？」**的な状態でインストールを終える事になります。

【審議中】「ｹﾞﾑ」?(´・ω・) (・ω・`) 「ｼﾞｪﾑ」ﾀﾞﾖﾈ?

・・・真にmikutterを楽しむにはgemの知識は割と必須で、学習機会の喪失と言う点ではあまりヨロシクない対応な気がしますが、オートメーション化の宿命だと思って諦めています。

(2)ショートカットアイコンの作成
デスクトップにみくったーちゃん印のショートカットを作成します。これをダブルクリックすれば、さながらWindowsネイティブアプリのようにmikutterが起動します。
（ちょっと前まではRuby起動時にちらっとコマンドプロンプトが出てダサかったのですが、ひこさん(@hico-horiuchi)が解決してくれました。）

ショートカットの作成とデスクトップフォルダの取得はWindows Scripting Host(WSH)用のActiveXをwin32oleモジュール経由で呼び出して実現しています。

また、ショートカットファイルにはWindows独自のアイコンファイル（*.ico）が必要になるため、ミクッターちゃんアイコン(mikutter.png)の変換を行っています。

Windows Vista辺りからアイコンファイルの中身にpngファイルがそのまま使えるようになったので、mikutter.pngにヘッダ情報を追加して即席でアイコンファイルを仕立てています。

5. おわりに

おわりに代えて、最後に一つ注意事項です。

本記事で色々書いたように、mikutter は Windows を公式にサポートしていません。現状は偶然が重なって何故か動いていると言うのが正しい認識です。

mikutter は日本語で開発に参加できるのも魅力の一つなのですが、非サポートの Windows に特有の問題を公式に報告してしまうと、無用な混乱が起こって mikutter 本体の進捗に支障を来す恐れがあります。

そのため、Windows で発生した問題は一旦私の Twitter アカウント（@moguno）または GitHub の Issue（https://github.com/moguno/mikutter-windows）までお願いします。ベストエフォートで直せる物は直しますので。

ではではー。

mikutter on Heroku

@firstspring1845

1. はじめに

mikutter はプラグインにより拡張可能なフレームワークとなっており GUI 等一般的な Twitter クライアントとしての機能もプラグインとして実装されています。(core/plugin 以下)これを削除してやることでフレームワーク部分を bot として使用することができます、今回は mikutter を Heroku で動かすことについていきます。

2. 環境の準備

mikutter のインストール自体については他の記事に書いてあったりするので省略します。適当なディレクトリに導入してください。

Heroku への登録及び gem を使用して Heroku のクライアントのインストールを済ませておいてください。またデプロイに必要なので Git をインストールし SSH の設定を済ませておいてください。

3. mikutter のデーモン化

@toshi_a さん曰く mikutter フレームワークを使って作られたのが Twitter クライアントの mikutter とのことですが、デーモンとして使うということは、もはや Twitter クライアントの mikutter ではないとのことなのでコンシューマーキーを取得し、core/config.rb を書き換えましょう。書き換えが終わったら起動して認証します。この時そのままだとユーザーのホームに認証情報が保存されるため mikutter.rb –confroot=conf のようにパスを渡してやることでローカルに認証情報を保存します。認証が終わったら core/plugin 内の streaming 以外の全ファイル及びディレクトリを削除し、streaming 内の spec ファイルの plugin 以下二行を plugin: []に書き換えます(gui を要求しているのでこうしないと読み込まれない)。

以上で mikutterd の準備は完了です。動かしたいプラグインを普通に導入しておいてください。

4. Heroku へのデプロイ

heroku login を実行し Heroku のアカウントにログインし heroku keys add (生成した公開鍵のパス)を実行し Heroku 側に公開鍵を登録します。

先ほど用意した mikutter のディレクトリに Procfile という名前のファイルを作成し bot: ruby mikutter.rb –confroot=conf と書いて保存します。git init を実行し Git リポジトリを初期化し、適当にコミットしておきます。

コミットが終わったら heroku create –stack を実行します。すると Heroku 側にリポジトリが作成されローカルの Git リポジトリと関連付けられます。git push heroku master を実行しローカルリポジトリの内容を Heroku 側に送り、heroku scale bot=1 を実行し有効化します。heroku logs もしくは heroku ps を実行してやれば動作しているかの確認ができます。

5. おわりに

私は以前自作の bot を Heroku で動かしていたものの設計ミスによりメンテナンスが大変でした。しかし、mikutter に移し替えて大幅にメンテナンスが楽になりました。何より mikutter プラグインとしてそのまま使えるのがとても良いです。

この記事はうろ覚えで書いている部分が多いので間違っている部分があるかも知れません。過去にも Heroku で mikutter を動かそうとした人の記事[1]があるのでそちらも参考にしてみて頂けると幸いです。

[1] https://gist.github.com/rinx/6361821

mikutterの見る夢

～何も言えなくて…夏～

ねぴぁ (@nepiadeath)

1 はじめに

私とmikutterとの出会いは2012年の6月まで遡ります．当時ほとんどtwitterとは縁のなかった私ですが，興味本位でmikutterを使いはじめるとすぐにその魅力に取り憑かれました．通知設定からサウンドを設定するとTLが動くたびに音が鳴って精神が浄化されていくというあのとき感じた喜びを今でも覚えています．

あれから2年が経ち，ついにmikutter3.0のリリースとなったわけですが，その新機能にあのときと同じような感動を覚えました．

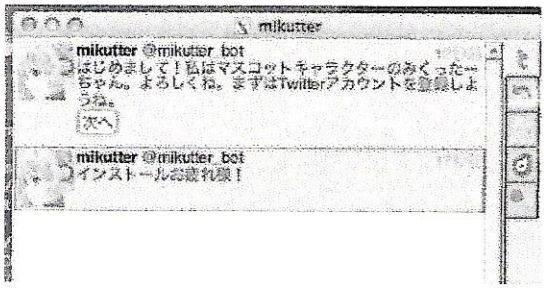


図1: 念願の夢・みくったーちゃんとの会話

なんとなんと！ あのみくったーちゃんと(一方的に語りかけられるだけの)会話を楽しむことができるのです！

しかし，やはりmikutterにはこれで満足してほしくはないものです．新たな課題を提起しつつ，mikutterの未来を考えるため，私は次のようなものを試みることにしました．

2 擬似的に喋るみくったーちゃん

mikutterの夢．それは，みくったーちゃんが「声で」喋ってくれること．今までも「ふぁぼっ」とか「りついーと！」など部分的には喋ってくれてはいました．今回試みるのは「会話」(システムメッセージ)の部分についてです．

とはいえ，時間も金も技術もないので(ポケミク買ってどうこうするのもキツい(つらい)．ということで，次のような方法で「みくったーちゃんが喋るとしたらこうなるんだろうなあ」程度のモデル実験を行ってみました．

~~(mikutter界ではタブーとされている)~~OSXにはsayコマンドというものが存在します．これを用いて擬似的にみくったーちゃんを喋らせてみることにしました．

2.1 sayコマンドについて

OSXのsayコマンドには複数の声があり，-vオプションで任意の声を指定できるようになっています．今回はみくったーちゃんを喋らせるという目的のためどこぞの外人に一ちゃんの声が出てしまっては元も子もないので，まずは女性の声で日本語を喋れる"Kyoko"を使えるようにしましょう．

```
$ say -v Kyoko みくったーちゃんだよー
```

うむむ…~~オバさんっぽい声だし~~まったくみくったーちゃん要素がないですが今回はプロトタイ

だと割り切ってこれを使いましょう.

2.2 mikutter_shaberu.rb

今回の実験のためのプラグインを作成します. ファイル名は"mikutter_shaberu.rb" とでもしておきましょう.

```
Plugin.create :mikutter_shaberu do
  on_appear do |ms|
    ms.each do |m|
      if m.user == "mikutter_bot" then
        sentences = m.message.to_s
        system("cat <<EOF | say -v Kyoko
              #{sentences}")
      end
    end
  end
end
```

これを plug'n ディレクトリに置きます.

2.3 mikutter を起動する

さて, 起動してみましょう.

すると…やはり先ほどの~~オバさんっぽい~~声でみくったーちゃんのセリフが再生されました!

再生されたとは言っても, いくつか問題点はあるようです. まず, 声が再生されている間は操作ができないこと. 次に, あまり長いセリフだと途中までで切れてしまう (?) こと. URL なんかは喋らずに飛ばしてしまってほしいですね.

3 mikutterの見る夢

今回はなんら大したことではなく, OSX の say コマンドを借りる形でみくったーちゃんを喋らせる, ということを試みてみました.

これは今後本当にみくったーちゃんが喋るようになることを期待しての課題の提起であり, 実用性を伴ったものではありませんでした.

先述したように私はポケミクを持っていませんが, ポケミクさえあれば本当にみくったーちゃんを喋らせることも可能でしょう. (実際としぁさんの mikutter_pocket_miku プラグインを拡張すればいけそう)

私も誰かがポケミクでみくったーちゃんを喋らせるプラグインをつくってくれればポケミクを買おうと思っています.

みくったーちゃんの声で起きて, みくったーちゃんの声に送り出され, みくったーちゃんの声でおかえりなさいと言われ…そんな風に, たのしい人生をておくれるようになりたいものですね.

わたし「きれいな声してたんだね…知らなかったよ…」

みくったーちゃん「(まだ声が出せないので) 何も言えなくて…夏」

みくったーちゃんちのGTK事情[1]

コスモ(@cosmo__)

1. はじめに

mikutterはRuby-GNOME2のgtk2というgemを使っています。これはRubyからGNOMEのgdk/gtkのバージョン2系を使えるようにする目的のgemです。
このgemは今はメンテナンスモードに入っていて、よほどの事が無い限り新機能が追加される事は有りません。もちろんバグフィックスやロード時のwarningには対応しています。

このgtk2 gemですが、RubyのC拡張を使って書かれています。

Ruby-GNOME2ではrcairo gemに依存しているgemが多く、また、Ruby-GNOME2のgemはほとんどglib2 gemに依存しています。

```
rcairo ← glib2 ←その他 Ruby-GNOME2のGNOME2に関係するgem
```

という依存関係が成立しています。

と、ここまではGNOMEの2系に関するgemに関してです。

2. Ruby-GNOME2のGNOME3系に関するgemについて

ここからが本題。現在、Ruby-GNOME2ではGNOME3系に対するgemの開発を進めています。Ruby-GNOME2では新たにgithub:kouさんがGNOMEにあるgbject-introspectionの仕組みに乗っかるgobject-introspection gemを開発しています。これにより、半分自動で新しいGNOME3系の機能が取り込まれるようになる予定です。

このgemの詳しい使い方はまだ書かれていないです。。。結構便利なgemだと思うんですけど。。。

3. mikutterのgdk3/gtk3への移行時期

mikutterをgdk3/gtk3に移行させるとしたらいつ？

たぶん、まだ早いです。

今のところ、gobject-introspection(以下giと表記)ベースになっているのは2.1.0リリース時点では

リリース済み

・clutter

1　編者改題。原題「mikutterに関連するGTK事情」

・clutter-gtk
・clutter-gst
・gio2
です。

未リリース
・gdk3
・gtk3
・gtksourceview3

となっていて、これはまだリリース出来る品質になっていないか、そもそもgiベースの実験的なgemのコードも書かれていない状態です。mikutterに関係するgemではそもそもgdk3/gtk3と考えられるのでかなり時期尚早という事が分かりますね。

~~mikutterからのフィードバックに期待しているので（？）是非ともgiベースになったらmikutter側でガシガシ使ってください！~~

4. Ruby-GNOME2でのテストコード

これは**mikutterを使っていたら気づいたらコミッタになってた**というKernel/VMでの発表のスライドにも少し書いてあります。

Rubyでのテストコードは最初のコードのユーザーになる、という意味合いが強いです。結構頑張ってテストコードがないコードをなくそうというRubyistは多いですね。コンパイラなどの静的解析のツールが無い以上コードを自動的にチェックするなら人間が頑張ってテストを書かないといけません。

Ruby-GNOME2ではテストコードを書くのが難しいという事情もあり、ほとんどテストが無いものも有ります。そして、すべてに書いてしまうとメンテナンスコストの増大や、開発が楽しめなくなるという問題も有ります。

giベースのgemではそのあたりのバランスが変わる事が予想されています。Ruby-GNOME2ではRubyらしい書き方が出来るGNOMEのバインディングを提供する、というプロジェクトの趣旨があります。

これに則ると、giベースのgemではRubyらしい書き方が出来る事を確認する意味合いのテストコードをメンテナンスするようになります。

メンテナンスコストに見合うテストコードを書くというように変わりつつあります。

5. そういえばMacでは？

うっ。これも出来る範囲で対応を進めています。

Mac で gdk3/gtk3 の gem をビルドする際には、brew コマンドがシステムにあるならば、gem をインストールする際に Homebrew の gtk+3 Formula を使う事ができます。既に Homebrew へ gi を有効化する Pull Request を何個か投げています。もし gdk3/gtk3 ベースに置き換えるとしても Homebrew 側の対応は終わっています。

今は gtk2 ベースの mikuter は動くんじゃないでしょうか！

mikutter を動かす事自体は難しくないものの、Mac-UIM を使った日本語入力周りで苦戦すると思います。

~~mikutter の開発者が自身の Mac で mikuter を動かせない時代なんてのもありましたね~~

6. まとめ

Ruby-GNOME2 では gtk2 gem は現在メンテナンスモードです。ほぼバグフィックスのみで新機能の追加はほとんど有りません。

gtk2 の後継の gem である gdk3/gtk3 は現在の人の手による実装から gi ベースの実装に置き換えている真っ最中です。また、gtk2 と違って gem が分かれている事に注意してください。

メソッド名やクラスが変わってしまっているので、gem の lib 以下に deprecated.rb が新たに作られていて、移行の手助けになるでしょう。

mikutter と niconico と、音楽のある生活

@shibafu528

前置き

皆さん、mikutter で Twitter ライフを満喫していますか？ただタイムラインを眺めるだけでなく、豊富な拡張性を活かしてプラグインを導入することでより快適な環境を構築したり、自らプラグインを書いたりしている方もいらっしゃると思います。

ところで話は変わりますが、mikutter という名前には某ミクさんが深く関わっていることと思います。実際に、付属するふぁぼ通知音や作者の@toshi_a 氏などを通してミクを感じることができます。しかし、mikutter では VOCALOID として活躍するミクさんの歌声を聴くことができないのです。

ならば、mikutter でミクさんの歌声を聴けるようにすればいい、本記事はそれを実現するための方法と開発したプラグインの紹介となります。

Nsen を聴こう！

ミクさんが初めに活躍した場所といえば、ニコニコ動画です。最近では全体のサービス名として niconico と名乗っていますね。ここには今も昔も、多くの VOCALOID オリジナル曲が投稿されています。

そして、このニコニコ動画では生放送サービスの一部として「Nsen」という動画配信放送が行われています。これは、ジャンルごとに視聴者のリクエストを受けつけ、抽選に当たった動画を配信していくものです。人気の動画のチェックや、自分の知らない埋もれた動画の発掘もできる、何もしなくても延々と動画を視聴し続けることができる~~自堕落促進~~作業用 BGM に適したサービスです。

もし、Nsen で配信されている動画の情報を取得することができれば、mikutter でも VOCALOID 楽曲を聴くことができますね？なんと、あまり手間をかけずに取得することが可能です。今回は、拙作のプラグイン mikutter_niconico に追記する形で実装を行いました。

Nsen では、生放送運営コマンドの形式で動画の情報を配信します。これらは全て、コメントのストリームに流れているため、ニコ生用のコメントビューアソフトで確認することができます。

コメントを取得するには、放送に紐付いたコメントサーバに接続しソケット通信を行います。ニコ生のコメントサーバに接続する手順を追ってみましょう。

まず、ニコニコ動画 API を使用するためにはログイン Cookie が必要です。「https://secure.nicovideo.jp/secure/login?site=niconico」に mail と password を乗せてアクセスするか、知識をお持ちであればブラウザのプロファイル等から取り出すといった方法で取得しておきます。

ログイン後、API「http://watch.live.nicovideo.jp/api/getplayerstatus/」を叩き、生放送の情報を取得します。「getplayerstatus/nsen/#{hoge}」というように Nsen のチャンネルを指定する必要がありますので、チャンネルと URL の対応は以下の表を参考にしてください。

Nsen チャンネルとリクエスト URL の対応表

ch01 VOCALOID	nsen/vocaloid
ch02 東方	nsen/toho
ch03 ニコニコインディーズ	nsen/niconicoindies
ch04 歌ってみた	nsen/sing
ch05 演奏してみた	nsen/play
ch06 PV	nsen/pv
ch99 蛍の光	nsen/hotaru
ch00 オールジャンル	nsen/allgenre

返ってきた XML 内の要素「ms」内からコメントサーバのアドレス(addr)・ポート番号(port)・スレッド ID(thread)を取得します。この辺は Mechanize gem を使うとお手軽です。

```
 <getplayerstatus status="ok" time="1401524687">
    :
    <ms>
       <addr>omsg104.live.nicovideo.jp</addr>
       <port>2808</port>
       <thread>1355336753</thread>
    </ms>
    :
</getplayerstatus>
```

取得したアドレスとポート番号を用いて、コメントサーバに接続します。そして、スレッド ID を含めたリクエストの文字列を送信します。以後、コメントをリアルタイムに取得することができます。

例えば、このように TCPSocket を使用して接続を行います。

```
TCPSocket.open(address, port) do |sock|
   sock.write("<thread thread=\"#{thread}\" version=\"20061206\"
res_from=\"0\"/>\0")
   while true
      stream = sock.gets("\0")
      # stream が受信した XML 形式のコメントデータ
   end
end
```

res_from に負の整数を与えると、その分の過去のコメントを取得することもできます。今回は、接続以降の運営コマンドのみを拾いたいだけで、過去ログは必要ないので 0 を指定しました。
これでコメントサーバへの接続とコメントの取得を行うことができました。要素の内容がそのままコメントの本文です。
このうち、/play と/prepare で始まるコメントが、それぞれ「動画の再生」「次の動画の事前ダウンロード」を指示するコマンドになっています。

```
/play smile:sm9 main "新・豪血寺一族 -煩悩解放 - レッツゴー！陰陽師"
/prepare sm9
```

/play コマンドでは「smile:」という文字の後に動画 ID が、/prepare コマンドでは単にスペースを空けた直後に書かれています。これを抽出することで「今 Nsen で流れ始めた動画」「次に Nsen で流れる動画」が分かるようになります。
なお/prepare コマンドは、基本的には/play コマンドの流れた直後に出現します。動画の再生を開始しつつ、次の動画について通知しているのだと推測できます。これを使うことで、次の動画について予め取得を行っておく処理を実装することもできます。

Nsen で流れる動画について情報を取得できるようになったので、次は動画を取得します。これもニコニコ動画 API を通して行うことができます。
(動画のタイトルや形式、再生回数などの詳細情報を取得するための API も存在していますが、今回はとりあえず動画ファイルを取れればいいので使用しません。)

使用する API は「http://flapi.nicovideo.jp/api/getflv/#{id}」というもので、動画 ID を渡すことで動画の保管 URL を取得することができます。但し、事前に対象の動画の watch ページを一度閲覧している必要があります。
例えば「flapi.nicovideo.jp/api/getflv/sm9」で以下のような結果が返ってきます。

```
thread_id=1173108780&l=319&url=http%3A%2F%2Fsmile-pcm42.nicovideo.jp%2Fsmile%3Fv%3D9.0468&link=http%3A%2F%2Fwww.smilevideo.jp%2Fview%2F9%2F9291984&ms=http%3A%2F%2Fmsg.nicovideo.jp%2F10%2Fapi%2F&ms_sub=http%3A%2F%2Fsub.msg.nicovideo.jp%2F10%2Fapi%2F&user_id=9291984&is_premium=1&nickname=%E8%8A%9D%E7%94%9F&time=1401790842626&done=true&hms=hiroba.nicovideo.jp&hmsp=2558&hmst=340&hmstk=1401790902.qSiYvWo7YA7_lOg7Jbbl6fNcTOE
```

結果は&区切りで、キー=値 の形式で記述されています。この中から url 要素を取り出して URL エンコードされた文字列をデコードすることで動画の URL を抽出することができます。この辺も Mechanize 等を駆使してパパッとやってしまいましょう。

```
# agent = Mechanize.new
agent.get("http://www.nicovideo.jp/watch/#{ID}")
response = agent.get("http://flapi.nicovideo.jp/api/getflv/#{ID}?as3=1")
flvinfo = {}
# レスポンスをハッシュに格納する
response.body.scan(/([^&]+)=([^&]*)/).each do |i|
   flvinfo[i[0]] = i[1]
end
# 動画を保存する
open(TITLE, "wb") do |f|
   # url を CGI::unescape でデコードしてから渡す
   f.print agent.get_file(CGI::unescape(flvinfo["url"]))
end
```

これで再生するための動画を確保できました。実際にプラグインとして使う時は、コメントサーバからの受信を行うスレッドを立てて、コマンドを受信したらダウンロードを行い、完了しだい再生を行うといった形になるでしょう。

GStreamer でマルチメディア再生

上記の作業によって、Nsen で流れている動画を調べてダウンロードを行うことはできました。ですが、このままでは mikutter ではこれまで音楽を聴くことが考慮されていなかったために大きな問題があります。「マルチメディアを再生できる仕組みが存在していない」のです。

音を鳴らすことに限定すれば、ビルトインの Sound プラグインや ALSA プラグインが既に存在しています。ですが、これらのプラグインはあくまで通知音を鳴らすためのものであり、途中停止や音量の設定といったメディアプレイヤーらしい事ができないものとなっています。

また、ALSA プラグインの内部で呼び出されている aplay コマンドでは、wav 形式のファイルしか再生することが出来ません。つまり、flv や mp4 でアップロードされている動画ファイルを直に再生することはできない、今回の目的を達成することはできないということです。

ですが、この目的を達成できるようなマルチメディアを扱うフレームワークが存在しています。その名は、GStreamer です。

*NIX 系環境を使っている方であれば、マルチメディア環境を整える際にもしかしたらパッケージ名等で見たことがあるかもしれませんね。

GStreamer は音声や映像といったマルチメディアを扱うための機能を提供している、クロスプラットフォームでオープンなライブラリです。また、Ruby-GNOME2 プロジェクトによって、Ruby バインディングが公開されているため、Ruby からでも使用することができます。

Ruby/GStreamer を使えば mikutter でもマルチメディア！……そう思っていたのですが、実際には mikutter 上で GStreamer を直接使用しようとすると再生中に処理が返ってこなくなるといった問題に見舞われました。
また、Ruby/GStreamer 2.2.0 時点では Ruby 2.1.0 上で動作させる際に Gst.init が呼び出されたタイミングで例外がスローされ使用できないバグが存在しています。既に修正のプルリクエストが受理されているため、次期バージョンでは問題なく動作すると思われますが執筆時点ではモンキーパッチを当てておくなどの工夫が必要となっています。

```
# -*- coding: utf-8 -*-
require "gst"

=begin
コード引用元:
https://github.com/ruby-gnome2/ruby-gnome2/issues/232
https://github.com/ruby-gnome2/ruby-
gnome2/commit/29dd9ccdf06b2fe7d9f5cf6ace886bb89adcebf2
=end

module Gst
   class Loader
      def call_init_function(repository, namespace)
         init_check = repository.find(namespace, "init_check")
         arguments = [
            1 + @init_arguments.size,
            [$0] + @init_arguments,
         ]
         succeeded, argc, argv, error = init_check.invoke(:arguments => arguments)
         succeeded, argv, error = init_check.invoke(:arguments => arguments)
         @init_arguments.replace(argv[1..-1])
         raise error unless succeeded
      end
   end
end
```

Ruby2.1.0 で動作させる際の例外を回避するモンキーパッチ

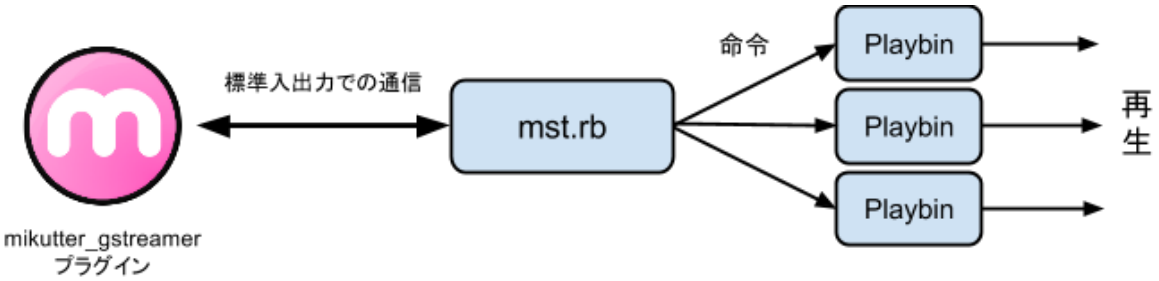
さて、こうなってしまったからには、mikutter プラグイン上に直接 GStreamer のライブラリを叩くコードを乗せることは難しいので、私は GStreamer ライブラリを操作するプログラムを挟み、mikutter からはそちらにプロセス間通信を行うことで解決しました。

“mst.rb”と名付けたこのプログラムは至って単純で、標準入力から命令を受け付け、その処理を行います。”play /path/to/file”とすればそのメディアファイルを再生し、”stop”とすれば停止する、そういったものです。
また、mst.rb では複数のファイルを同時に再生することも実現しています。命令の最後に識別子を書き加えることで、並行して再生するようになります。

mst.rb 内では再生識別子ごとに fork してプロセスを生成、その上で GStreamer ライブラリのメソッドを呼び出しています。わざわざ子プロセスにしているのは、単一の GStreamer パイプラインに制御が奪われることを防止するためです。
結果として、mst.rb のプロセスから更に子プロセスとパイプ通信を行うコードとなってしまいましたが、思ったよりは複雑化していないので私個人としては良しとしました。

子プロセス上の Gst パイプラインでは、コードを平易にするために Playbin エレメントを使用しています。これは、ファイルや URI からの入力・メディアの形式に合わせたデコーダの選択・サウンドデバイスへの出力を一纏めにしたコンポーネントです。URI 形式でファイルを指定するだけですぐに再生が行えることと、音量設定などを行えるようにもなっていることが採用のポイントです。

以上の実装をまとめ、GStreamer 再生プラグインでは以下のような形で mikutter からサウンドを再生することとなります。



こうして遠まわしに利用することによって、mikutter から GStreamer が使えない問題を解決することができました。

ここまでしなくても、もう少し別のアプローチを考えて mikutter 上でできないものか、そう考える方も居ると思います。私もそう思ったこともありますが、不安のあるコードを mikutter から完全に分離することで TL を巻き込んでクラッシュしにくくなると考えると、この手法は面倒でもそこまで悪くはないと思います。

後は前項の方法によって Nsen から取得し保存した動画ファイルのパスを、GStreamer プラグインに渡すだけで、再生ができるようになります。

おわりに

少しばかり手間をかけて、音楽を入手し続けるソースと mikutter でメディアを再生する環境を整えることで、ついに mikutter でもミクさんを始めとする VOCALOID の方々の歌声を聴けるようになりました。
特に GStreamer 周りは私のような初心者が触れるには闇が深く、幾多のトライ&エラーの繰り返しの結果がこのような形となりました。もっと賢い方法があったら是非とも教えていただきたいと考えています。

今回の GStreamer サウンドプラグインの実装は Github にて公開しています。音楽再生だけでなく、defsound に引っかけることで通知音の再生にも対応させています。

これで mikutter というソフトウェア名に合うような、みっくみくな環境を作るお手伝いが少しはできたと自負しています。

とあるておくれはこう言っていたような気がします。mikutter は環境である、と。これを文面そのままに捉えるのであれば、Twitter だけに捕らわれず最高のみっくみく環境を作ろうとすることに意義はあると思いました。mikutter 楽しい。

プラグインについて

記事で取り上げた自作プラグインは Github 上に公開しています。
niconico: https://github.com/shibafu528/mikutter_niconico
gstreamer: https://github.com/shibafu528/mikutter_gstreamer

mikutter 3 一巡り

@toshi_a

1. ごあいさつ

待望の mikutter 3 が遂にリリースされました。今回は、 mikutter 3 で新しく追加された機能を、サードパーティプラグインから使える機能を中心に見て行きましょう。

1.1. 本稿の見方

用語の種類によって書式を変えています。

event	mikutter コアやプラグインで定義される変数、クラス、イベント等
Array	Ruby や外部ライブラリで提供されるクラス等
コマンド	mikutter 独自の用語

2. イベントヒント

イベントを明示的に宣言する手段が導入されました。これはまだ実験的な機能ですが、将来的にはイベントの最適化のための重要な機能になってくるかもしれません。

```
defevent :favorite,
  priority: :ui_favorited,
  prototype: [Service, User, Message]
```

例の場合、 **favorite** イベントを定義しています。 **favorite** イベントは、あるツイートがお気に入りに追加された時に発生します。

2.1. priority

priority	**意味**
ui_response	UI に関係する、ユーザの入力がトリガーとなって発生し、イベントが発生した結果、画面上にフィードバックを表示しうるイベント
routine_active	UI に直接関係しないイベントで、割り込み実行する必要のあるイベント
ui_passive	UI に関係するイベントだが、ユーザの入力がトリガーになっていないもの
routine_passive	(デフォルト)
ui_favorite	ふぁぼ

Delayer キュー[1] にイベントが入る時の優先順位です。同じ優先順位なら通常のキューですが、優先順位が高いものがより早く実行されます。

Delayer キュー に沢山のイベントが登録されている時に新たに UI 関連のイベント（例えばプロフィール表示）が発生すると、残ったイベントを全部実行してから処理を開始することになるので、例えばタブを開くのに 1 秒以上かかるということがありました。これを解決するには、先にプロフィールを開くためのイベントを実行してやればいいのです。

現状では、ふぁぼイベントの優先順位を下げるために使われているくらいです。ふぁぼイベントは 1 秒間に 10 イベント発生する可能性があり、ちょっとでも mikutter の動作がもたつくと一気に *Delayer キュー* を圧迫することがあります。そうすると、大量のふぁぼのせいで他のイベントの実行が遅くなり、結果としてユーザは動作が重いと感じます。優先度を下げることで、ほかのジョブがあれば常にそちらが優先されるので、ユーザの操作が滞ることが少なくなります。

ふぁぼを遅延させるといっても、何もこの変更によって数秒も遅れるわけではありません。むしろ、ふぁぼ爆撃を受けている間も変わらず動き続けることは、快適にふぁぼ爆撃を受けることにつながります。もちろん、ふぁぼ以外についても同じことが言えます。

2.2. prototype

イベントに期待する引数の個数と、振る舞いを定義します。 例のように Class や Module が指定されたら、is_a 関係にない値でイベントを発火しようとした場合例外が発生します。また、 Symbol のインスタンスの場合はそのメソッドを持っているかどうかをチェックします。

イベント周りでは、明らかに間違った値でイベントを発火すると、発火側ではなく受け取り側がクラッシュするという問題があります。バックトレースはクラッシュした場所しか教えてくれないので、原因となったイベント名くらいしかわかりません。 期待する値のインターフェイスを表明しておくことで、 **Plugin#call** を呼んだ時に、その場で例外が発生します。イベントをフックする側はインターフェイスが表明されているから、間違った値が渡される心配をする必要がなくなります。

3. 異なるスレッドでフィルタを実行

以下の条件を満たすとき、フィルタがメインスレッド以外で動作するようになりました。

- boot イベント発生後
- イベントの発生によってフィルタが呼ばれた場合

例えば次のコードは、test イベントが発生すると、パラメータをフィルタが実行されたスレッドで置き換えます。

1 mikutter で発生するイベントを集めてあとで実行する仕組み。3.0 からは Delayer の mikutter への依存が解消され、外部ライブラリに切り出された https://rubygems.org/gems/delayer

```
Plugin.create(:test) do
  filter_test do |x|
    [Thread.current]
  end

  on_test do |x|
    p x
  end
end
```

出力は、例えば **#<Thread:0x007f12618b5e50 run>** になります。この時の Thread.main の結果は **#<Thread:0x00000001c4a620 run>** なので、フィルタが別のスレッドで走ったことがわかります。

具体的には、 **Plugin#call** が呼ばれた時に **SerialThread** に登録され、フィルタの実行が終わり次第 Delayer に登録されます。フィルタがないイベントの場合は、直接 Delayer に登録されます。フィルタは、ものによっては処理に非常に長い時間がかかるものがあり、時間がかかるとフリーズしたように見えます。そこで 3.0 では実験的に、フィルタを別のスレッドで実行して、処理に時間がかかったら一度中断してメインスレッドに処理が移るようにしました。

ただし良いことばかりではなく、フィルタとその他が別のスレッドで走るということはつまり、スレッドの同期問題を考えなければなりません。また Ruby では、別のスレッドを使ったところで単純に処理速度が上がるというものでもありません。しかし、もともとフィルタというのは **Plugin.filtering** で直接呼べばそのスレッドで実行されてしまうので、実行されるスレッドは不定なのです。この辺りについては、 mikutter の薄い本 vol.3 の私の記事「mikutter フレームワーク」の「並列実行と非同期処理」[2]で詳しく述べているので、併せて参照してください。

4. 抽出タブのデータソースを拡張

3.0 から、抽出タブが大幅にパワーアップしました。UI から設定できる機能も増えましたが、今回からはデータソースを追加することができるようになりました。このことによって、プラグインが追加したデータソースを、ユーザがフィルタしたり統合したりして利用できるようになったのです。

```
filter_extract_datasources do |ds|
  ds[:teokure] = "ておくれ".freeze
  [ds]
end
```

extract_datasources フィルタで、 **teokure** という識別名を持った「ておくれ」というデータソースが定義されました。プラグイン側から使うのが前者で、後者はユーザに表示される名前です。

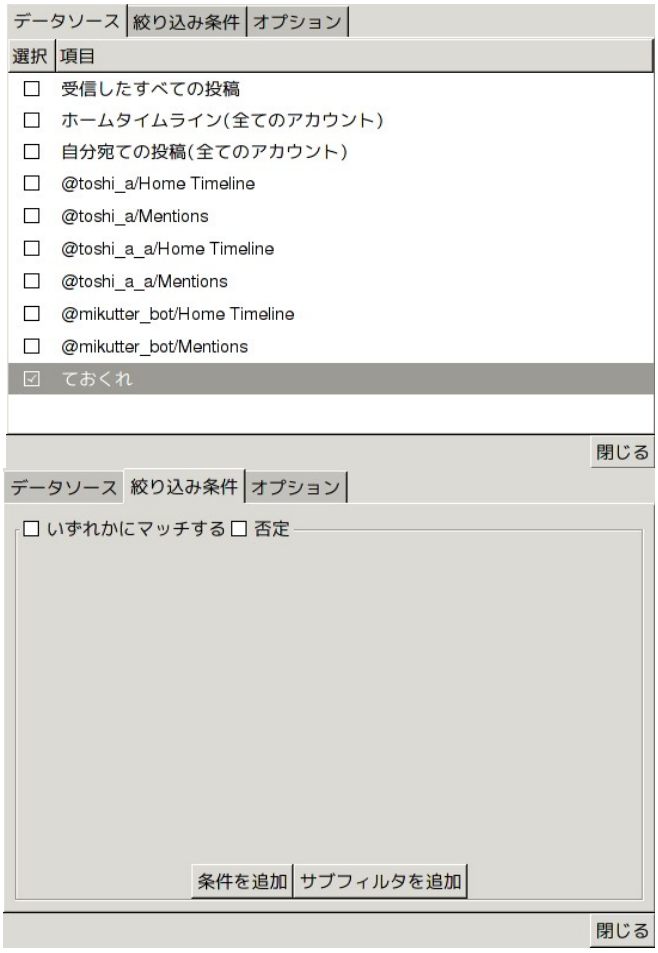
最後に、このデータソースにデータを注入する処理です。

2 mikutter の薄い本 vol.3 『人類は手遅れました』 http://kohinata.sumomo.ne.jp/mikutter.html

```
on_appear do |ms|
  filtered = ms.select do |m|
    m.to_s.include? "ておくれ".freeze
  end
  unless filtered.empty?
    Plugin.call :extract_receive_message, :teokure, filtered
  end
end
```

appear イベントに流れてくるすべてのツイートから「ておくれ」を含むツイートだけを選んで、 **extract_receive_message** イベントを発火します。これで、 **teokure** データソースを使っている抽出タブにこのツイートが注入されます。これでもう使用できるので、実際に抽出タブを作ります。

設定画面を見ると、早速「ておくれ」データソースができています。



フィルタは今回はいじりません。何もしなければ「すべて」になります。

これで誰かが「ておくれ」とツイートしたら、それがこのタブに入ってきます。尤も、単なるキーワードマッチなら抽出タブの設定でできてしまうのですが。

この機能は、例えば学習機能のある複雑なフィルタを実装した時に、データソースとして公開することで、ユーザはそれを組み合わせて TL を構築できることを意味します。データソースにしておけば、TL の表示を実装する必要がありませんし、ユーザはそのタブを消したり増やしたり、新着通知を表示することもできるのです。ゆくゆくは、ホーム TL やメンション等も抽出タブを用いた実装に変えたいと思っています。

5. 多言語プラグイン

mikutter 3.0 からは多言語対応が入りました。言語ファイルは、プラグイン毎に管理されています。当然、サードパーティプラグインが言語ファイルを用意して同梱することで、ロケールに応じて UI の言語を切り替えることができます。

5.1. 下準備

以下のコマンドで spec ファイル(.mikutter.yml)を作成します(パスと mikutter の位置は適宜読み替えてください)。

```
$ mikutter spec ~/.mikutter/plugin/teokure
```

あとは .mikutter.yml を編集し、依存プラグインに **uitranslator** を追加してください。

```
---
slug: :teokure
depends:
  mikutter: 3.0.0
  plugin:
  - uitranslator
version: '1.0'
author: toshi_a
name: teokure
description: teokure datasource
```

後述の makepot は、このプラグインに依存しているプラグインに対してのみ働きます。

5.2. 多言語対応コード

試しに、先ほどのておくれデータソースプラグインを多言語対応にしてみましょう。以下のようになります。

```
filter_extract_datasources do |ds|
  ds[:teokure] = _("ておくれ".freeze)
  [ds]
end
```

Plugin#_ は、そのプラグインで、引数の文字列を現在のロケールの言語に対応した文字列を返します。

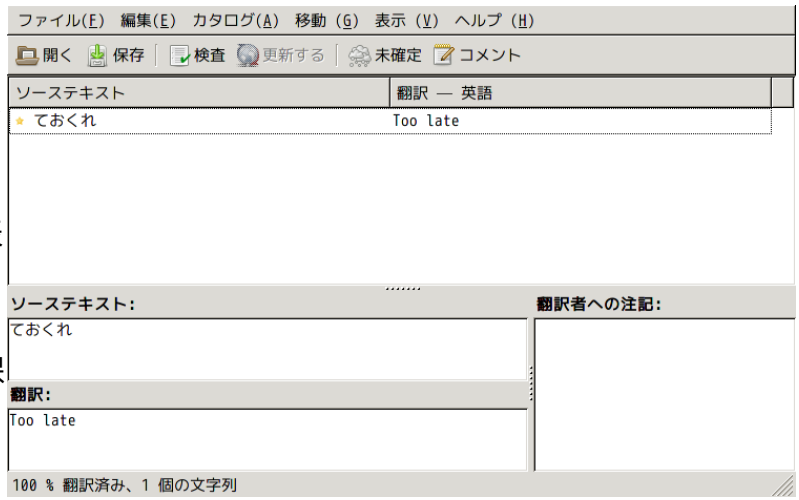
5.3. pot ファイルの作成

以下のコマンドを実行すると、teokure プラグインの pot ファイルが生成されます。

```
$ mikutter makepot teokure
mkdir -p /home/toshi/.mikutter/plugin/teokure/po
mkdir -p /home/toshi/.mikutter/plugin/teokure/po/ja
$ ls ~/.mikutter/plugin/teokure/po
ja/  teokure.pot
```

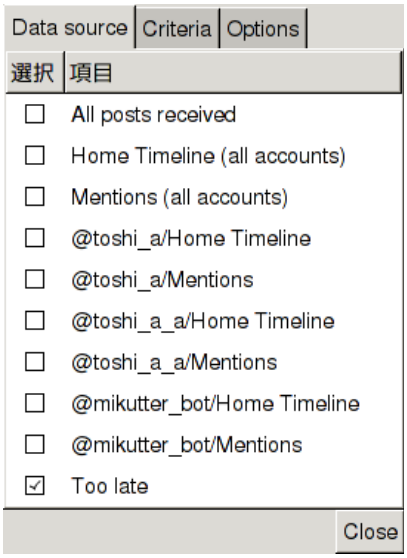
5.4. 翻訳

生成された言語ファイルを編集します。翻訳には専用のエディタを使うと便利です。説明は割愛しますが、例として GUI の言語ファイルエディタ poedit[3] を使用すると、右のように翻訳候補が表示されています。



例では英語なので、翻訳したファイルは po/en/teokure.po に保存します。あとはその言語で mikutter を起動します。

```
$ LC_ALL=en_US.UTF-8 mikutter
```



抽出タブの設定で、データソース一覧を見ると、「ておくれ」が「Too late」になっていると思います[4]。

翻訳は、継続的にやっていくのはモチベーションの維持がなかなかむずかしく、また上で説明したとおり、訳を考える以外の作業も多いことが、余計に敷居を上げています。ただ、mikutter 自体のネームバリューが上がってきたり、Linux で使用できる Twitter クライアントの中で世界的に有力であったと思われる「Hotot[5]」が開発を停止したりしたことも影響してか、mikutter の外国人ユーザが出始めているのは事実で、より多くの人に自分のプラグインを使ってもらいたい！と考えている人は、プラグインの他言語化も考えてみてはいかがでしょうか。

6. まとめ

mikutter3.0 で入った変更のなかでも、プラグイン開発者にしか関係がない変更をいくつか取り上げました。多言語対応は今後これをしようとする人には貴重な情報源となるでしょう。また、抽出タブは mikutter のツイート管理の方法を大幅に変えうるプラグインです。

最近では特にプラグイン開発者が増え、今までになかった全く新しいプラグインも出てきています。そういったプラグインをより作りやすくなるように mikutter を成長させていければ、と思っています。

3 http://poedit.net/
4 タブとか閉じるボタンもちゃんと英語になって…ｳﾜｰｯ！（多言語化忘れが写り込んで爆発四散）
5 http://www.hotot.org/

＜＜後書＞＞ @brsywe

働くのって大変ですね。

さて今回もておくれなかった mikutter の薄い本『ておくれ！えもーしょん』　いかがでしたでしょうか。
書きたいことはいろいろあった気もしますが締切が近いのと（原稿出すのが遅れて）サークルの後輩さんをいじめるのもよくないと思っていることもあってここらへんで。というか後書きって何書いたらええんやろ。わからなくなってきたのでわたしは ALIA's CARNIVAL!に戻るね。

・イラストなど提供
表紙ロゴ「ておくれ！えもーしょん」 @T_H_sister
裏表紙みくった！！　@shijin_cmpb
まいどありがとうございます。

――次号予告と寄稿者募集――

原稿提出期限：12 月 5 日

頒布予定：コミックマーケット 87,こみトレ 25

問合せ先：@brsywe , @ch_print

奥付
発行日：2014 年 9 月 15 日
PDF 版
発行：mikutter の薄い本制作委員会
発行者： @brsywe 西端の放送局内喫茶室長
連絡先： brsywe　@　hotmail.co.jp

ご意見・ご感想は Amazon ギフト券のメッセージ欄にどうぞ。

mikutter の薄い本制作委員会ウェブページ
http://kohinata.sumomo.ne.jp/mikutter.html

mikutter の薄い本制作委員会では、Amazonギフト券・つり銭の寄付による金銭面の支援を受け付けております。もし、あなたがこの薄い本を読んで、何かしら満足感を得られたなら、支援して貰えるとその満足感を誰かと共有できるかも。上のウェブページを御参照ください。

※PDF版のみ、18, 19ページの記事は、提出された時から編集できないPDFファイルであったため印刷してスキャンをせざる得ず、不鮮明ですが仕様です。