mikutter

の薄い本 vol.9

いいね♡が捗りすぎてヤバい。

都会から離れたておくれた *TL* に暮らす主人公・としぁ。

歳の近いておくれは、妹[1]のそらまめだけ。

彼の家系は「ておくれ力」と呼ばれる不思議な能力を備えた、ておくれの血を引く一族だった。

京大過ぎるその力を京都府警には知られてはいけない――それが一族の掟なのである。

目次

イラスト

1 という設定。

2 冊子版から開くページをかえました。

前書にもなれなかった

@brsywe

1. 待ち人来たり

前号 mikutter の薄い本 vol.8『テオクレズガーデン』の前書きでも触れた SobaCha の薄い本がコミックマーケット 89 でついに出ているそうである。

2. 今回の mikutter の薄い本

1 : totori.dip.jp で提供されている API についての @misodengaku による詳説である。今頃紅茶を飲んでいるはず。
2 : ドイツのトリと化した@Asyley_によるドイツのトリライセンス解説。ぜひ 600eur.gochiusa.net/ も参照されたい。
3 :@osa_k 。今までに作った頭のおかしい、実用的なプラグインについて。
4 : @Akkiesoft。 σ(o'v'o)まとも。
5 : @toshi_a 。

イワシがいっぱいだあ...

ちょっとだけもらっていこうかな

6 : @shijin_cmpb。これについてはちょっと解説を。

3. トトミミハイエース

すべての元凶は @toshi_a と @brsywe の酔いだ。vol.8 発刊時に病床に伏していた @toshi_a の退院祝いでの席である。

https://twitter.com/shijin_cmpb/status/642774591350292481

というわけで今回はトトミミハイエースが掲載された。えっちだが 18 禁とすべき理由も見当たらないので X-rated とならず。

4. まとめ

希里乃ちゃんかわいい

totoridipjp API 1.1 ドキュメント

@misodengaku

はじめに

この文章は totori.dip.jp で提供している API のドキュメントです。totoridipjp API を使用することで Web ページのスクレイピング等が不要となり、簡単にトトリちゃん画像を呼び出したりすることができます。

プログラミング言語によってはライブラリを用いることでさらに簡単に扱うことが出来ます。いくつかの言語に関して以下に示します。

- Ruby: totoridipjp　gem https://github.com/toshia/totoridipjp-ruby
- Java: totoridipjp4J　https://github.com/wakamesoba98/totoridipjp4J

1.0 との差異

同一メジャーバージョン内では上位互換を保つため、新機能を使用しない場合は新バージョンの使用に際して特に改修は必要ありません。

変更
/api/topimg/に神棚経由モードが追加され、トトリちゃんフィギュアの下を通ったパケットを得ることが可能になりました。
追加
以下のエンドポイントが追加されました。

- /api/buy_now/
- /api/pages/

エンドポイント毎の解説

/api/topimg/

トップページのトトリちゃん画像への URL を取得する

指定できる GET パラメータ

format

json, xml, text のいずれかを指定します。それ以外はエラーとなります。

指定しない場合は json 形式で返ります。

god

1 を指定した場合神棚経由モードとなります。1 以外を指定した場合は神棚経由モードにはなりません。

神棚経由モードではトトリちゃんフィギュアの載った RTX1210 経由で totoridipjp 運営者の自宅サーバ(旧 totoridipjp サーバ)から画像を取得します。

パケットがフィギュアの下を通過することが目的のため、キャッシュを無効化するような HTTP ヘッダを付加します。

回線の上り速度が最大 10Mbps であるため必要時以外は 1 を指定しないでください。

使用例

http://totori.dip.jp/api/topimg/

トトリちゃん画像への URL を JSON 形式で取得します。

http://totori.dip.jp/api/topimg/?format=text

トトリちゃん画像への URL のみをテキスト形式で取得します。

/api/buy_now/

トップページ内「買え」の内容を取得する

指定できる GET パラメータ

format

json, xml のいずれかを指定します。それ以外はエラーとなります。

指定しない場合は json 形式で返ります。

/api/pages/

トップページ内、トトリちゃん画像と「買え」の間にあるリンクのリストを取得する

指定できる GET パラメータ

format

json, xml のいずれかを指定します。それ以外はエラーとなります。

指定しない場合は json 形式で返ります。

relative

1 を指定するとレスポンス内の url に http://totori.dip.jp/を/とした相対パス表記が混じります。指定されなかった場合、全ての url は絶対パス表記になります。

これは明示的に絶対パスへの変換動作を停止させるためのオプションです。通常必要ありません。

ドイツのトリと学ぶ「クリエイティブ・コモンズ」

@Asyley_

1．はじめに

ドイツのトリ生誕 n＋1 周年を迎えた今年、鳥類を創作対象とする動きが多く見られた。

「ドイツのトリ」は筆者がドイツで撮影[1]した UMA の一種であるが、Twitter 上に投稿されただけのその撮影画像について、Twitter 外で利用・創作等する場合には、「撮影者に許可がいるか」「改変してもいいか」「商用 OK？」など、個人のモラルによってしまうのが現状である。

これに伴い、ドイツのトリは「クリエイティブ・コモンズ[2]（以下 CC）」にて利用条件を明確に設定して提供することとした。筆者も知識が非常に***ておくれ***ているため、今回はドイツのトリを用いた CC 利用について、わずか 2P で読者と共に学ぶことで、mikutter の薄い本 vol.9 の”前座”とする。

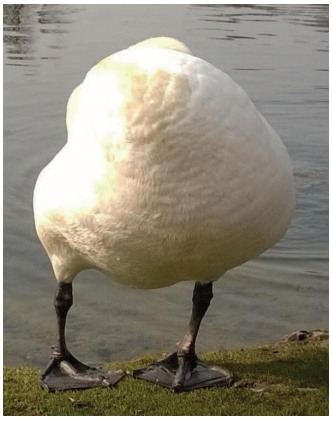


図 1　ドイツのトリ

2．CC ライセンスとは

作品に対して、利用条件を設定して使用を認めるものである。CC にはその条件のキツさに種類があり、ドイツのトリは最も緩い条件である「表示のみ」にて提供した。図 2 に示すマークを作品に添えることで宣言することができる。

以下が「表示のみ」で提供された作品の利用条件である。

1）提供元のクレジットを明記すること

これだけである。これさえ守れば、改変、商用利用等を自由に行って良い。条件の厳しい CC ライセンスはこれに加え、改変、商用利用等を制限したものとなる。

図 2　CC ライセンス（表示のみ）[2,3]

3．運用

偶然にも筆者監修の元、***初音ミク***（@3qgt）さんの所有する Web ページ[4]にてドイツのトリの画像の公開、および CC ライセンスマークの表示依頼

を快諾いただいたので、ここを”ドイツのトリ公式”として CC ライセンスでの提供を開始した。

当該サイトはシンプルな構成であり、CC 提供にあたり、サイト内に設置した要素は以下のとおりである。

1）ドイツのトリの画像（作品）
2）CC ライセンスのマーク
(クリックで利用条件の説明ページへ飛ぶようにリンクした)
3）CC にて提供されている旨の文面

任意だが HTML に「作品のタイトル」「作者名」等の詳細なメタデータを書けるように準備されており、書けばより円滑な CC 提供が可能となるが、ここでは割愛する。

4．活用しよう

これにて「クレジット書けば誰にでも好き勝手改変して金儲けの足しにできる（誰かはよ作って）鳥類」の完成である。

しかしながら、筆者の努力も虚しく正式な活用事例はまだほとんど無い。***おこ***[5]である。(と思ったが最近活用事例があった[6]）結局、前回の C88 では誰もドイツのトリグッズを作ってくれないので自分で制作して金儲けの足しにした。

5．まとめ

とは言うものの、多所でドイツのトリをネタにして頂いているのは感謝である。ここまで非常に多くを割愛しつつ最低限の説明であったが、ドイツのトリに限らず、これを機に CC を活用した創作活動において参考の 1 つとしていただけたら筆者として嬉しく思う。

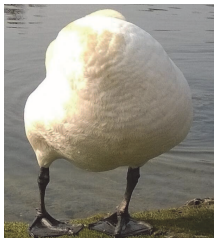
6．参考資料・文献

[1] 座標：緯度 48.173915, 経度 11.554186
[2] http://creativecommons.org
[3] Creative Commons (CC BY 4.0)
[4] http://600eur.gochiusa.net
[5] 日本のネットスラング
[6] http://fleurdulap.in

著者情報

あしゅりー（@Asyley_）

ぷに！あしゅりーだよ。じ、自分のサークルの新譜がまた大変なことになっちゃったぷに！＞＜も、もう怒ったぷによ……今日はもう絶対ぷ



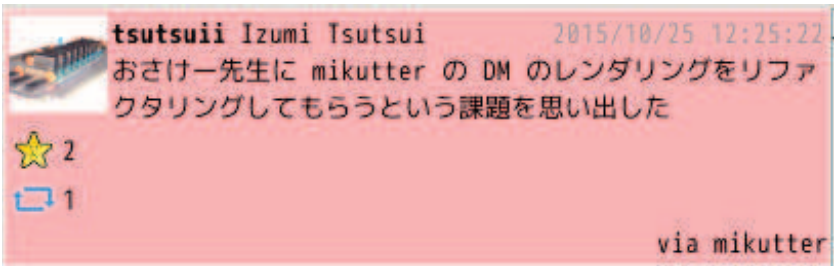
ドイツのトリ

ギョエエエエエアアア！！！ギョェエェエエエエ！！！！ギャアアア！！！！！！ギャギャァアァ！！！！！

今まで作った mikutter plugin

@osa_k

こんにちは、@osa_k です。OSC2015 Tokyo/Fall でいつものように mikutter ブースの展示をしていたら、@tsutsuii さんの



というTweetが流れてきたので[1]、これを機に1年くらい放置してしまっていたDMのレンダリング改修とかをやって記事にしようかと目論んでいたのですが、就職して以来30億のデバイスで動く某言語に魂を捧げてしまったために全然進みませんでした。もうしわけねぇもうしわけねぇヽ('ω')ノ三ヽ('ω')ノ。せっかくの薄い本なのにまた記事を落としてしまうのももったいないので[2]、今回は今まで作ったプラグインを振り返ってみようかと思います。順不同でいきます。

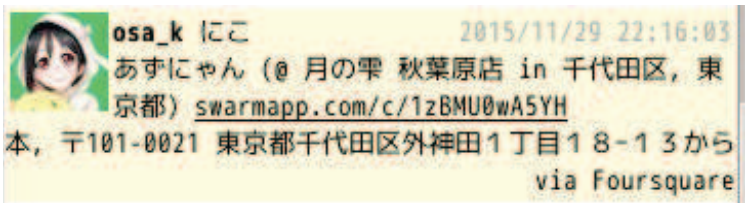
shindanmaker (https://github.com/osak/shindanmaker)

診断メーカーのURLをクリックすると、投稿窓に診断結果を入力します。最近は診断メーカー人気も下火になってきていて使ってる人もそんなにいないかなぁと思っていたのですが、11月頭に診断メーカーがhttps化したときにはすぐ@zwxi さんからPull Requestを頂いて、ちょっと嬉しかったです。

当初は診断結果用の投稿窓は増えず、最初からある投稿窓に診断結果を貼り付けるだけだったのですが、@toshi_a さんのパッチによって現在の形になりました。mikutterの投稿窓周辺のコードは結構複雑になっていて、この機能が追加された後もいくつかバグを見つけたりしました。

geocode (https://github.com/osak/geocode)

位置情報付きのツイートが流れてきたとき、緯度と経度をGoogleのGeocode APIに投げて地名に変換し、SubPartsとして表示します。確か台湾に行ったときに、せっかく位置情報付きのツイートをして自慢しても mikutter から見えなくて寂しかったので、帰ってきてから Advent Calendar のネタも兼ねて作ったプラグインだったと思います。



Geocode APIは容赦ない長さの文字列を返してくることが多く、よくはみ出します。ちゃんと処理すれば対処できると思いますが、あまり困っていないので放置されています。このプラグインを作っている時にSubPartsのheightを

1 背景が赤いのは、@pengui2716 さんの favorited_count プラグインの効果です。累計ふぁぼ数が100を超えると赤くなります。
2 mikutterの薄い本 vol.6『ておくれ！えもーしょん』に参加登録したとき、修士の闇に呑まれて記事を書けませんでした。

間違えて指定していたかなんかで、1 ツイートが異常に縦長になった挙句、メモリをものすごい勢いで食い尽くして PC がフリーズし、グラフィック周りのコードは難しいなぁと思いました。

usercomplete (https://github.com/osak/usercomplete) & userutil (https://github.com/osak/userutil)

投稿窓でユーザ名を補完するためのプラグインです。Twicca の補完が便利だったので mikutter にもあるといいなあと思って作りました。このプラグインは 2013 年頃に作ったのですが、当時はみっくストアを初めとしてプラグイン周りの機能拡充が盛り上がっており、そういった状況の影響もあって、mikutter プラグインとしては珍しい 2 モジュール構成（ユーザ補完のメイン部分は userutil プラグインで行い、usercomplete は投稿窓での UI を実装する）となっています。プラグイン管理ツールが発展すれば 2 モジュール構成でも普通に使えるし問題ないだろうと当時は思っていたのですが、実際のところは 2015 年が終わろうとしている今になってもこれといった管理ツールの決め手はなく、ちょっと導入が面倒になってしまっています。

```
>>> Plugin.filtering(:user_prefix_search, 'toshi_a', [])
["toshi_a", [User(@toshi_a_kanyou), User(@toshi_a_bot), User(@toshi_a_)
```

userutil は他のプラグインから利用可能なフィルタを提供することだけに焦点を絞っており、GUI プラグイン等には依存していません。@penguin2716 さんの mikutter_shell_post プラグインで導入される@whois 機能[3]と組み合わせると、特定ユーザのプロフィールをすぐに見ることができて便利です。

show_tweet (https://github.com/osak/show_tweet)

Tweet の URL や ID からそのツイートを取得してきて、新しいタブに表示するだけのプラグインです。Twitter が何回も仕様変更したり、そもそも Entity 等の仕様が明確でなかったりすることもあり、mikutter は Tweet をレンダリングする処理に結構バグがあったりするのですが（URL のリンクとして機能する場所が見た目と大きくずれていたり、レンダリングに失敗して落ちたり）、大抵の場合こういった問題は特定の Tweet を表示したときに発生するので、再現する手段があるとデバッグが大変やりやすくなります。そう考えると mikutter 本体の機能にあってもおかしくないと思うのですが、なぜか存在していないので作りました。

右下にある Tw と描かれたアイコンをクリックするとダイアログが出てくるので、Tweet の URL や ID を入力して OK を押すと会話スレッドと同様のタブでその Tweet が表示されます。

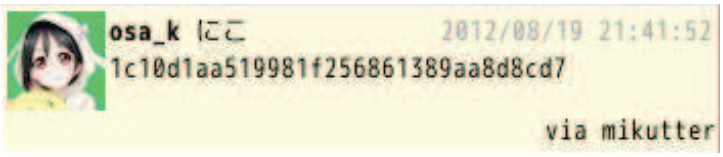
eject (https://github.com/osak/mikutter-eject)

2013 年の 1 月に、なぜか突然色々な言語・環境で Eject するのが流行り始めたので便乗しました。しかし Eject の第一人者として有名な@Akkiesoft さんはこの遥か前から Eject されてたんですね。すごい。機能としては、上の画像にある Eject っぽいアイコンをクリックすると CD トレイが排出されるものです。最初は単に Eject するだけでしたが、なぜか Pull Request をもらってしまい、eject するデバイスが指定できたり、複数回ボタンを押すと CD トレイの状態をトグルするようになっていたりします。

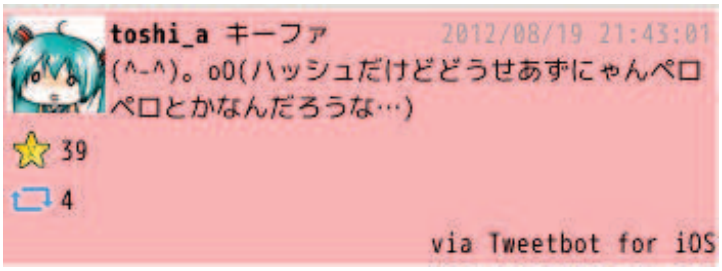
3 @whois @toshi_a のような形式で投稿をすると、実際にメッセージを投稿する代わりに指定したユーザのプロフィールを表示してくれる機能です。

md5tweet (https://github.com/osak/md5tweet)

とある人が、Twitter で公に言ってはまずいような愚痴をそのまま Tweet するのではなく、MD5 を計算して Tweet するという画期的な手法を使っていて感銘を受けたので作りました。



ちなみに@toshi_a さんは MD5 から元データがわかる能力持ちなので[4]、こういうことをしても解読されます。



まとめ

次回は技術的な話を書けるようにがんばります。はーあずにゃん

4 https://twitter.com/toshi_a/status/34507685214822400

ｱﾝｯ！sible で mikutter を ﾌﾞﾘｯ！ﾌﾞﾘｯ！ﾌﾞﾘｯ！ﾌﾞﾘｯ！

～mikutter をインストールする Ansible playbook をつくってみる～

あっきぃ（@Akkiesoft）

こんにちは、あっきぃです。わたしはσ(o'v'o)まともです。

本稿では、Ansible を利用して mikutter を導入する playbook を作ってみたお話を紹介します。

Ansible とは

Ansible は、Chef や Puppet などと同じ構成管理ツールのひとつです。構成情報(Playbook)の記述は、yaml ファイルでシンプルに記述することができます。また、展開するターゲットになるマシンとの接続には SSH を利用するため、マシンにエージェント等を用意する必要がない点も特長です。本稿のようなシンプルなものから、OpenStack のように大きなシステムの構築まで、幅広く利用されています。

また、2015 年 10 月には Ansible が Red Hat 社に買収されたことでも話題になりました。

Ansible の導入

Ansible は Python で記述されており、pip コマンドで簡単に導入することができます。Python と pip コマンドの導入の導入は各自検索してください（許して）。

```
$ sudo pip install ansible
```

mikutter playbook について

mikutter playbook は、mikutter の新規インストールとアップデートができる playbook で、GitHub 上で公開してあります。

https://github.com/Akkiesoft/RPi-tools

リポジトリ名からも分かる通り、Raspberry Pi を使う上で便利なものをまとめて置いているリポジトリになります。mikutter playbook はデモ環境の構築用として作成しました。そのため、基本的には Raspberry Pi の Raspbian、もしくは Debian 環境で利用されることが前提となっています。ただし、mikutter playbook のうち、Jessie と Ruby2.1 のロールを改善するか、あらかじめ Ruby と Bundler を手動でインストールしたうえで mikutter のロールだけを実行することで、それ以外の環境でも使えるようになるはずです。

mikutter playbook の用意と基本的な実行方法

まじめに解説をしようとすると、Ansible の使い方について延々解説する記事になってしまうため、ここでは Ansible の使用方法を最小限にとどめつつ、mikutter playbook を実行する方法を解説します。

まずは、mikutter playbookが含まれている先ほどのRPi-toolsリポジトリをGitHubからcloneします。

```
$ git clone https://github.com/Akkiesoft/RPi-tools
```

RPi-tools/ansibleディレクトリに移動して、hostsファイルを作成します。以下の例では、teokurepiをplaybook実行対象のホスト名とします。指定したホストは、実際に名前でpingやSSHなどの通信が行える必要があります。

```
[all]
teokurepi
```

そして、ansible-playbookコマンドを使用して、mikutter playbookを実行します。-uはSSHログインに使用するユーザー名です。あらかじめ指定のユーザー名で鍵認証ログインができるようにしておいてください。

```
$ ansible-playbook -i hosts -u pi mikutter.yml
```

hostsファイルに複数のホストを書き込むと、複数のホストに対して同時にmikutterのインストールが実行することができます。もし、hostsにたくさんホスト書いておきつつ特定のホストだけ実行したい場合には、ansible-playbookコマンドの-lオプションでホスト名を指定すると、実行対象のホストを絞り込むことができます。

コマンドを実行してSSH接続に成功すると、Ansibleはplaybookの構成情報にそってmikutterインストールのタスクを実行していきます。すべてのタスクの実行が終わると、最後にタスクの実行結果の集計が表示されます。

なお、初回実行時は、bundlerの実行タスクがあり、コンパイルなどの処理にとても時間がかかります。特にRaspberry Piで実行される方は注意してください。

mikutter playbookのポイント

このplaybookでは、mikutter APIを利用して最新バージョンを取得しています。これにより、最新バージョンの新規導入に加えて、実行対象のホストにmikutterが存在していてバージョンアップが可能であれば、バックアップをしたうえで新しいバージョンを導入するようなplaybookが実現できました。

http://mikutter.hachune.net/api

ぶっちゃけると、本稿mikutter API活用しているということを書きたかっただけなので、Ansibleでのmikutter APIの使用例をリスト1に示します。APIをJSON形式で取得して、バージョン番号を変数として格納します。

導入済みのmikutterのバージョン確認

には、mikutter 3.2.10 で導入された --version オプションを使用します（この記事のためにパッチを投げました)。これ以前 mikutter には--version オプションがありませんので、変数にはあらかじめ 3.2.9 を代入しておき、--version オプションが使用できたら、取得したバージョン番号で変数を上書きします。

バージョン番号の比較には version_compare と言う jinja2 フィルターを使用しています（リスト 2)。比較結果を install_flag に格納することで、以降のインストールを行うタスクの条件分岐を「when: install_flag」もしくは、「when: not install_flag」のようにシンプルに記述できます。

アップデートとして playbook が実行される場合には、オリジナルの mikutter ディレクトリーは「mikutter_<Playbook 実行時間>」と言う形式でリネームされます。

リスト 1: Ansible で mikutter API を利用する例

```
# mikutter API から JSON 形式でバージョンを取得する
- name: check latest version of mikutter
  uri: url="http://mikutter.hachune.net/download/unstable.json?count=1"
  register: mikutter_json
  always_run: yes
# 取得に失敗していた場合は playbook の実行を中止する
- fail: msg="can't get latest mikutter version!"
  when: mikutter_json.json[0].version_string is not defined
# バージョンナンバーを mikutter_ver に格納する
- set_fact:
    mikutter_ver: "{{mikutter_json.json[0].version_string}}"
```

リスト 2: バージョンの比較方法

```
# バージョンを比較して、結果を install_flag に格納
- name: compare version
  set_fact:
    install_flag: "{{ exist_mikutter_ver[1] | version_compare(mikutter_ver, '<') }}"
```

おわりに

Ansible Playbook 作りは、一通り作った後に調べ返すと、次々とスマートにやるテクニックが出てくるので、mikutter playbook でもだいぶいじくりまわしました。仕事でもいつもレビューで袋叩きに遭っているので、まだ改良の余地があるような気がしてなりません。

先述の通り、今のところは Debian 系での実行が前提になっていますが、今後はある程度他のディストリビューションでも実行できるように改良を加えていくつもりです。Ubuntu と、CentOS と、Arch Linux くらい…かなあ(ﾟ∀ﾟ)。

おまけ: teocraft 活動報告

本稿を書いている間にも、隙あらば Minecraft をしていました。Minecraft サイコー！！！

✌('ω'✌)三✌('ω')✌三(✌'ω')✌

ここ最近は teocraft での活動に集中しているので隙間埋めに報告してみます。

現在、私が teocraft で進行中の事業は、ラビットハウスネザーゲート付近に建設したゾンビピッグマントラップタワー整備です。以下の動画を参考にしています。

http://nico.ms/sm17593066
http://nico.ms/sm19717098
http://nico.ms/sm20418469

参考動画のように周辺を徹底的に掘り抜いて湧き潰しするのはだいぶ厳しいので、空洞の埋め立てとハーフブロック敷設による湧き潰しを行っています。それでも対象範囲はとてつもなく広くなり、見えない空洞も多々存在するので、動画のような効率で金インゴッドを得るのは難しいようです。湧き潰しの道は長い……。

図　ゾンビピッグマントラップタワー

手作り mikutter

@toshi_a

1. はじめに

今年に入ってからの多忙のためか、2 月頃から著しく体調が優れなかったのですが、ついに 7 月 21 日に病気であることがわかり、丁度一ヶ月間、入院しておりました。手術直前に mikutter の薄い本 vol.8 に遺稿を寄稿し、わちゃわちゃされ、心配をお掛けしました。今では社会復帰できて、調子に乗って **totoridipjp-ruby** などのキチ Gem を公開できるくらいには回復しました。ありがとうございます。

この **totoridipjp-ruby** というのは、あの有名な totori.dip.jp Web API の Ruby バインディングですが、この他にも mikutter 3.3 は以下の私が書いたライブラリに依存しています。

- Pluggaloid
- Delayer Deferred

特に Pluggaloid は、もともと mikutter の中核を担っているコードをライブラリにしたものです。ライブラリにするにあたって多少の変更はありましたが、当然 mikutter での使用に耐えるもので、mikutter 3.3 では mikutter が Pluggaloid を利用するようになりました。

今回は Pluggaloid とはどういうものなのか、実際に使用しながら紹介したいと思います。

1.1. Pluggaloid とは

mikutter は、プラグイン同士がイベントを用いて連携し、柔軟に機能を拡張できることが特長です。つまり mikutter 本体は、プラグインを作成する機能を提供し、イベントを送受信する機能を提供している…と考える人がいてもおかしくはありません。実際、Pluggaloid が出るまではそれは正しかったのです。Pluggaloid とは、まさにその部分を mikutter から削ぎ落とす試みだったのです。もはや mikutter3.3 にプラグインの仕組みは残っていません。

ライブラリになったということは、mikutter 以外のアプリケーションが、mikutter のプラグイン機能を利用できるということにほかなりません。今回は Minecraft Server の標準入出力をイベントによって抽象化するスクリプトを書いて、それをプラグインで拡張してみましょう。

2. mikutter を作る

2.1. bundler で Pluggaloid などをインストールする

まず、コードを書くディレクトリを作って Gemfile を設置します。コードは一応紙面にも掲載しますが、Github に公開しておくので実際に動かしてみたい人はそちらをご覧ください[1]。

```
# Gemfile
source 'https://rubygems.org'

group :default do
  gem 'pluggaloid'
end
```

これで、Pluggaloid と、それが依存する **Delayer**、**Instance Storage** などのライブラリがインストールできます。

```
$ bundle install --path=vendor/bundle
```

2.2. イベントループを書く

```
# app.rb
require 'bundler/setup'
Bundler.require(:default)

# Pluggaloid が使う Delayer は priority が必要
Delayer.default = Delayer.generate_class(priority: %i<high normal low>,
default: :normal)

require_relative 'plugin'

# イベントループ
loop do
  Delayer.run
  sleep 0.05
end
```

最初の 2 行で Gemfile に書いたすべてのライブラリをロードしています。require を書きまくる必要がなくていいですね。

イベントループも単純です。 `Delayer.run` というのは、Delayer キューにあるジョブを実行するメソッドです。イベントが発生した時なんかは必ず Delayer を介するので、この部分がメインループとなるわけです。0.05 秒というのは、Minecraft の 1tick の長さと揃えてみただけです。これといった意味はありませんが、あまり長く取り過ぎると反応がちょっと遅くなります。用途によってはそれでもいいんですが。

デフォルト Delayer を設定の部分は、ややこしいし上に本筋とは関係ないですが軽く触れておきます。Delayer クラスは **Delayer.generate_class** でいくつでも作成できます。クラスを作るのです。つまり戻り値は Class です。このクラスを複数作れば、例えば他の

1 https://github.com/toshia/miku_craft/tree/mikutter-book-v9

スレッドで実行される Delayer みたいなのを作れます。Proc しか入れることが出来ない Queue だと考えることもできます。複数存在できるということは、これを利用する Pluggaloid などに使用する Delayer を伝える必要があります。その一番簡単な方法が、デフォルト Delayer に指定することなのです。

さて、これだけでは起動したら Ctrl+C を押すまで戻ってこないだけのスクリプトで、mikutter の風上にも置けません。そこで、ここでロードされている plugin.rb です。残りの一切のことは plugin.rb に書いてあるのです。

2.3. Pluggaloid を初期化

```
# plugin.rb
include Pluggaloid

Dir.glob(File.join(__dir__, 'plugin', '*.rb')) do |ruby|
  require_relative ruby
end
```

なんとこれだけ。最初の include は、Pluggaloid をトップレベルで使えるようにするためのものです。これを書かないと `Pluggaloid::Plugin` みたいに参照しなくてはなりません。その後はとくに言うまでもないですが、plugin/以下のすべての Ruby ファイルをロードしています。ここに Ruby ファイルを置けばプラグインとして機能します。Pluggaloid 自体には外部のプラグインファイルをロードする機能はないので、ここは書く必要があります。

mikutter ではここで、サブディレクトリ内を探したり、 **.mikutter.yml** ファイルから定義ファイルを読み取って依存関係を考慮したりしています。さて、機能的には mikutter の劣化版ですが、これで Pluggaloid を利用した、手作り mikutter が完成しました。この mikutter がどの程度のものなのか評価するには、やはりプラグインを書くのが一番です。

3. 早速プラグインを書く

以上でプラグインを動かす機構はすべて整いました。早速プラグインを書いていきましょう。

3.1. minecraft プラグイン

いきなりやばげな名前のプラグインを実装します。そもそも今回は Minecraft Server の標準入出力を Pluggaloid で wrap し、プラグインでいろんな機能を足していきたいと思っていたのでした。mikutter 的に考えれば、その部分すらプラグインにしてしまうのは当然です。

```
# plugin/minecraft.rb
require 'open3'

Plugin.create :minecraft do
  mc_stdin = nil
  on_boot_server do
    mc_stdin, mc_stdout, mc_stderr, wait_thr = Open3.popen3('java -Xmx1024M
-Xms512M -jar minecraft_server.jar nogui')

    stdout_thread = Thread.new do
      mc_stdout.each do |res|
        Plugin.call(:server_raw_output, :stdout, res) end end

    stderr_thread = Thread.new do
      mc_stderr.each do |res|
        Plugin.call(:server_raw_output, :stderr, res) end end
  end

  on_minecraft_execute do |user_name, command|
    Plugin.call :minecraft_run_command, "execute #{user_name} ~ ~ ~ #{command}"
  end

  on_minecraft_give_item do |user_name, item_name, count|
    Plugin.call :minecraft_run_command, "give #{user_name} #{item_name}
#{count}"
  end

  on_minecraft_tell do |user_name, message|
    Plugin.call :minecraft_run_command, "tell #{user_name} #{message}"
  end

  on_minecraft_run_command do |command|
    mc_stdin.puts command
  end

  on_server_raw_output do |pipe, line|
    puts "#{pipe}: #{line}"
    case line
    when %r<\A\[\d{2}:\d{2}:\d{2}\] \[Server thread/INFO\]: (\w+) joined the
game\Z>
      Plugin.call(:join_player, $1)
    when %r<\A\[\d{2}:\d{2}:\d{2}\] \[Server thread/INFO\]: (\w+) left the
game\Z>
      Plugin.call(:left_player, $1)
    when %r<\A\[\d{2}:\d{2}:\d{2}\] \[Server thread/INFO\]: (\w+) fell from a
high place\Z>
      Plugin.call(:die, $1)
    end
  end

  Plugin.call(:boot_server)
end
```

これは本体よりも長くなりました。どのようなイベントが提供されているかだけ見てみましょう。当然、ここで宣言されるイベントは他のプラグインと連携するためのものです。

boot_server

Minecraft Server の起動です。後に一度だけ発生します。Minecraft Server が stdout や stderr に何か出力したら、 **server_raw_output** イベントを発生させます。

minecraft_execute

Minecraft の execute コマンドを実行します。

minecraft_give_item

Minecraft の give コマンドを実行します。

minecraft_tell

Minecraft の tell コマンドを実行します。

minecraft_run_command

Minecraft Server でコマンドを実行します。上 3 つのイベントは内部でこれを呼び出しています。

server_raw_output

Minecraft Server が出力した文字列を引数に発生するイベントです。実際には以下のイベントも発生するので、通常はそれを利用します。

join_player

プレイヤーがログインした時に発生します。

left_player

プレイヤーがログアウトした時に発生します。

die

プレイヤーが死亡した時に発生します。

minecraft プラグインは、子プロセスとして minecraft.jar を起動します。見た目には普通にサーバを起動したのと変わらないのですが、内部では上記のようなイベントが飛び交っています。早速それらを利用して、Minecraft のワールドに作用するプラグインを書いてみましょう。

3.2. 死んだら煽る

Minecraft では様々な理由でプレイヤーの HP が減少し、0 になると死にます。minecraft プラグインでは、落下死した時に **die** イベントが発生するようになっています。

それでは早速、落下死した人を祝福する花火を上げるプラグインを実装しましょう。

```
# plugin/meshiuma.rb
Plugin.create :meshiuma do
  on_die do |name|
    Plugin.call :minecraft_execute, name, "summon FireworksRocketEntity ~ ~ ~
{LifeTime:20,FireworksItem:{id:401,Count:1,tag:{Fireworks:{Explosions:
[{Type:4,Flicker:0,Trail:0,Colors:[16712965,16774912],FadeColors:[16777215]},
{Type:2,Flicker:0,Trail:0,Colors:[16777215],FadeColors:[16777215]}]}}}}"
  end
end
```

花火を打ち上げるには、 summon コマンドを使用します。 summon はエンティティを出現させるコマンドで、 FireworksRocketEntity が打ち上がっている花火のエンティティ名です。

summon コマンドは、エンティティを出現させる時に、特定のエンティティの場所を指定することができませんが、今回は Player エンティティと同じ場所に FireworksRocketEntity を出現させたいです。そういう場合には execute コマンドを使えば、特定のエンティティの場所を基準にしてコマンドを実行できます。minecraft プラグインでは、 **minecraft_execute** というイベントを呼び出すとこれが使えます。

エンティティの種類を変えてやれば何でも出せるので、死んだ場所に雷を落とす、透明なクリーパーを出現させてアイテム回収に来た人をアイテムごと吹き飛ばすなど、いろいろなことに応用可能です。

3.3. 通算ログインボーナス

さて、実は死んだ時に花火を上げる程度なら、マイクラ内でコマンドブロックを使ってもほぼ同じことを実現できます。マイクラ上で起こった出来事をトリガーにして Ruby のコードを実行できるメリットは、複雑な計算や、ファイルシステムなどとの連携です。今回は、サーバで長く遊んでいると良いものがもらえる通算ログインボーナスシステムを作ってみましょう。

3.3.1. 実装方針

ここで、ソーシャルゲームなどでよく耳にする「通算ログインボーナス」についておさらいしておきます。これはログインすること自体にインセンティブを発生させるための仕組みで、今までログインした日数を記録しておいて、ログイン時に一日一回、日数に応じて報酬を与えるものです。同じ日に複数回ログインしてもカウントされません。また、どんなに間隔が開いていても日数が減ったりリセットされたりすることはありません。

その他の仕様は以下の通り。

1 日の境目は 0 時とする

タイムゾーンはサーバに依存します。

サーバにログインした時に報酬の計算・付与を行う

丁寧にやるなら、ログイン時以外にも、0 時の時点でログインしている人全員にそ

の日の報酬を配布すべきですが、今回は行いません。

3.3.2. 実装

コマンドブロックだと現在の時間がとれないためこのようなことは難しいですが、Ruby なら簡単です。各ユーザの通算ログイン日数は適当に Marshalize して保存しておきましょう。mikutter なら **Plugin#at** などが使えるところですが、今回は直接ファイルを読み書きすることになります。

```
# plugin/daily_present.rb
Plugin.create :daily_present do
  save_file = File.join(__dir__, 'count.dat')
  counter = FileTest.exist?(save_file) ? Marshal.load(File.open(save_file,
&:read)) : {}

  on_join_player do |name|
    counter[name] ||= {last: Date.today - 1, count: 0}
    if Date.today != counter[name][:last]
      counter[name][:last] = Date.today
      counter[name][:count] += 1
      Plugin.call(:give_daily_present, name, counter[name][:count])
    end
    File.open(save_file, 'wb') do |out|
      Marshal.dump(counter, out)
    end
  end

  on_give_daily_present do |name, days|
    Plugin.call(:minecraft_tell, name, "ログイン#{days}日目！棒をプレゼント")
    Plugin.call(:minecraft_give_item, name, 'minecraft:stick', 1)
    if (days % 7) == 0
      Plugin.call(:minecraft_tell, name, "#{days}日記念！鉄をプレゼント")
      Plugin.call(:minecraft_give_item, name, 'minecraft:iron_ingot', 1)
    end
    if (days % 30) == 0
      Plugin.call(:minecraft_tell, name, "#{days}日記念！ダイヤのクワをプレゼント")
      Plugin.call(:minecraft_give_item, name, 'minecraft:diamond_hoe', 1)
    end
  end
end
```

ログイン日数などは `counter` に格納された Hash に保持します。また、 `save_file` というファイルにその内容を保存します。

ログインした時に現在の日数などを計算するための処理は `on_join_player` ブロックに書いています。このイベントはプレーヤーがログインした時に発生するのでしたね。ファイルにはログイン日数のほかに、前回ログインした日付が入っているので、今日最初のログインかどうかが判断できます。このブロックはその日最初のログインだった場合に `give_daily_present` を発生させるところまでです。

`on_give_daily_present` には、景品を渡すコードを単純に書いています。毎日ログインするだけで棒、7 日ごとに鉄インゴット、30 日毎にダイヤのクワが貰えます。

4. Pluggaloid とは

4.1. mikutter との互換性

以上作成したプラグインと、プラグインを動かすための小さなスクリプトで、Minecraft をホストとする全く新しいアプリケーションを書くことが出来ました。プラグイン部分はコードをすべて貼り付けましたが、長々と書いた割にはさして目新しい内容ではないです。というのも、これらは mikutter も使っている Pluggaloid を使っているだけですから、mikutter プラグインでもあるわけです。これらのプラグインを ~/.mikutter/plugin/ 以下に設置すると、mikutter が Minecraft を起動することになります。何も修正は必要ないでしょう。

4.2. Pluggaloid の未来

Pluggaloid があれば mikutter が無くてもここまでできる、ということを確認しました。これは確かに良いことですが、一方で mikutter でこの程度のことは出来るということでもあります。Pluggaloid の存在にどういうメリットがあるのでしょうか。

4.2.1. Pluggaloid の質が上がることで mikutter の質も上がる

mikutter は 2015 年 12 月 25 日で 6 周年を迎えました。その多くのコードは 5 年前に書かれており、この 5 年での私の成長を考えれば、見るに耐えない品質です。mikutter 自体もある程度大きくなってきて、思わぬところで関係なさそうな複数のコードが依存していることもあって、メンテナンスコストが上がっていくばかりです。 そこで、mikutter 3.0 くらいから、Delayer をはじめ、いくつかのコードを mikutter から切り離し始めました。こうすると、目的が一つになるのでメンテナンスがしやすいというメリットもありますし、再利用可能になるという副次的な効果もあります。

4.2.2. 必要な機能だけを使える

今回作成した Minecraft にいろいろするスクリプトは、Twitter との連携もないので mikutter を使う必要はありません。こういう時には mikutter を使うのではなく、Pluggaloid を利用するのが良いかもしれません。mikutter の機能を使いたいときは、mikutter を使えば良いのです。逆は難しいですが、mikutter への乗り換えは容易でしょう。

4.2.3. Minecraft MOD としての可能性

このスクリプトが Minecraft に対してできることは案外少なくて、その原因のひとつに Minecraft と対話することが意外と難しいという点があります。例えば、あるプレイヤーがブロックに変更を加えた時に何かしたいというような要件には向きません。

そこを解決したければサーバ用 MOD や Bukkit Plugin を開発し、mruby で Pluggaloid

を動かすということも視野に入ってくるでしょう。プラグインを書いてある程度自由にMinecraftを拡張できると考えると、なかなかおもしろそうなテーマではあります。しかし、今まで見てきたようなことがたったこれだけの労力でできるのであれば、これはこういうものと割りきって便利に使っていくというのでも十分だと思います。MODと違って、Minecraftのバージョンアップに強いというメリットもあります。

4.2.4. 暇だった

あること無いこといろいろ書きましたが、入院中暇だったからやったというのが一番の動機です。諸事情で病室ではノートPCを使いたくなかったのでmikutterを利用できる環境がなく、自宅にsshでログインして気が向いた時にやっていました。 そして退院後にmikutterとくっつけてみて、うまくいったというわけです。今年のアクティビティの大半を入院中のコードが占めているので、これが一応の成功を収めてよかったと思います。

5. まとめ

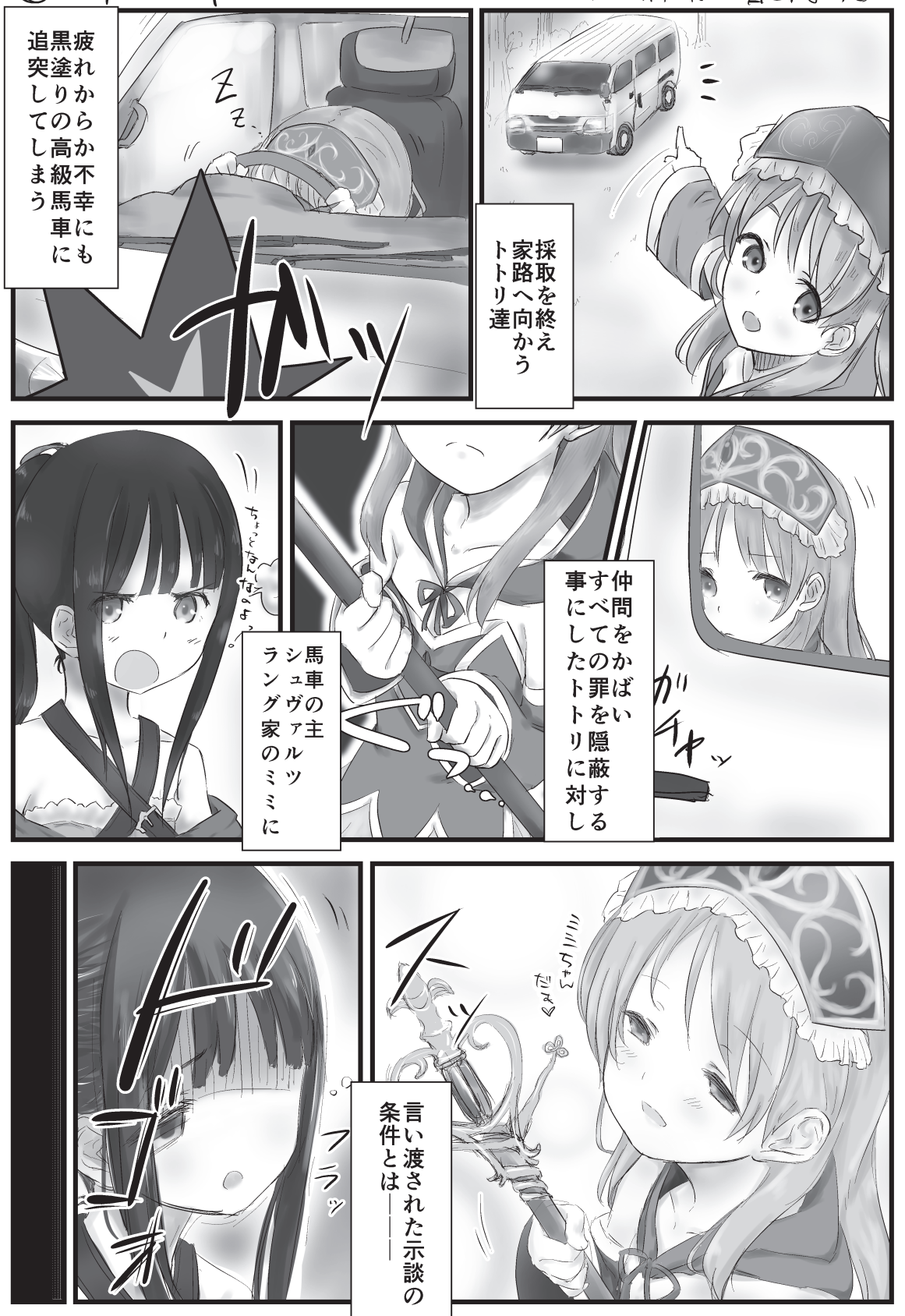
今回はPluggaloidを単体で使う例として、Minecraftと通信してみました。

mikutterは、Pluggaloidを利用してTwitterクライアントやデーモンを実現している、と表現することもできます。ならば今回作ってきたものは、Pluggaloidを利用してMinecraft Serverを拡張したと考えることができます。大げさな対比に見えるかもしれませんが、嘘ではないですね。

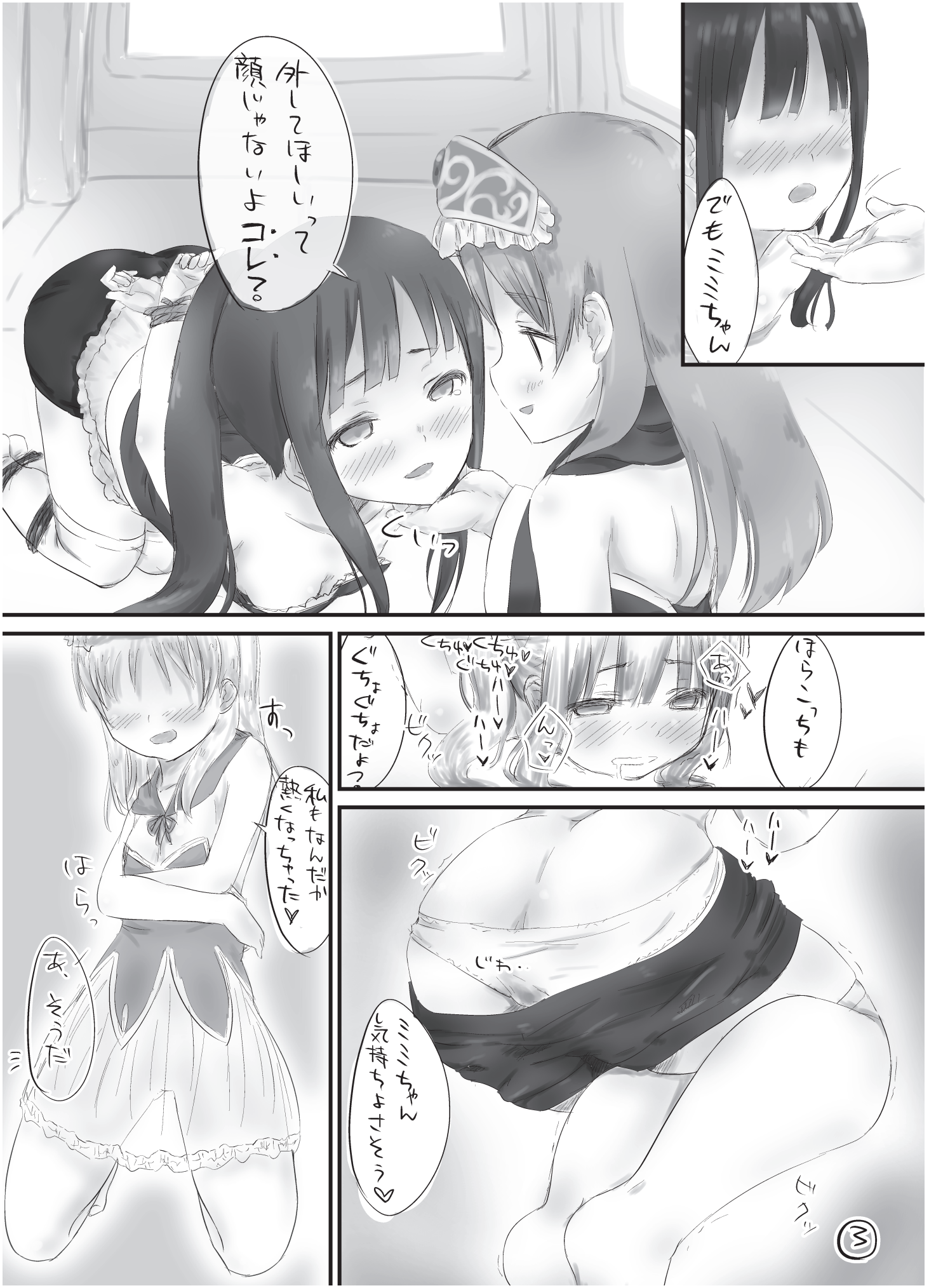
具体的な例としてこういったものを作ってみたのですが、当初思っていたよりは実用的なものができたので満足しています。本稿からはカットしましたが、最初は『ログインしました』『ログアウトしました』とputsするだけのプラグインを紹介する予定だったのですが、次々に思いつくアイデアにまかせて手を動かしているうちに、より実用的なログインボーナスへと置き換わりました。こういういきあたりばったりな思いつきで非生産的なことをしている時間に一番価値があると再認識しました。まさにmikutterですね。

@shijin_cmpb

※本作はキャラ崩壊を含みます。




ギチッ
!?
な、
何よコレ…!?
ハ
ミミちゃん起きたっ。
おはよ
トトリっ。
びっくりした？
かわいい
は、早くコレ外しなさいよ
ギッ
ん～～？
ビクッ
本当に外してほしいの？
にっ
あ、当たり前でしょ


でもミミちゃん
外してほしいって顔じゃないよコレ。
くいっ
ほらここも
あっ
ハーハー
んっ
ぐちょぐちょだよっ。
ビクッ
ハーハー
ビクッ
じわ..
ミミちゃん
いい気持ちよさそう♡
ビクッ
3
す、
私もなんだか熱くなっちゃった♡
ほらっ
あ、そうだ


おクスリ
くいっ
ここ
ちゅっ♡
ん♡
ぷはっ
ぷちゃ♡
ぴちゃ♡
ぬちゃ♡
ビクッ♡
ね
もっともっと
気持ちよくなって？
ミミミちゃん♡
はっ♡
あっ♡
んっ♡
んっ♡
あ♡
ダメ♡
トド♡
ビクッ♡
あ♡
あ♡
あ♡
あ♡
くちゅ♡
くちゅ♡
きゅっ
ビクッ
まだまだ、
もっとボクのこと
気持ちよくなってね♡
4

＜＜あとがき＞＞

強い精神

――次号予告と寄稿者募集――

原稿提出期限：6 月 30 日

頒布予定：コミックマーケット 90

次でいよいよ 10 巻目ですよ。なんか記念になるものが欲しいよね。

問合せ先：@brsywe , @ch_print

奥付

発行日：2015 年 12 月 31 日(コミックマーケット 89)：冊子初版第一刷
2016 年 1 月 31 日（こみっくトレジャー27）
2016 年 2 月 20 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

印刷・企画・編集協力：IUJK (special thanks: @aeoe39950426)

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。