peercast技術合同誌

Broadcast yourself

IPv6とつながらないP2P

CS:GO サーバー改造事始め

VRC民に効く(かもしれない)
Unity操作Tipsまとめ

Misreading Chat
リスニングガイド

センテンス指向
プログラミングの提案

目次

はじめに

この本は PeerCast コミュニティの技術合同誌です。

PeerCast っていうのはオープンソースの P2P 動画配信ソフトなんですが、そのユーザーが集まって技術関係の記事を載せました。

ただ PeerCast のユーザーが書いたってだけで、この本の内容に PeerCast 自体はほとんど関係ないので、PeerCast 全然知らんという人でも安心してお読みいただけます。逆に PeerCast についてはなーんにも書いてませんので、もし PeerCast について知りたければ別途調べてみましょう。

IPv6 とつながらない P2P

著 :kumaryu

おうちで IPv6 を使うことと、IPv6 で (IPv4 でも)P2P がなかなかつながらないことを解説します。

おうちで使うことを想定しているので、IPv6 についてよくわからない人向けです。

1. IPv6 とは

IPv6 ていうのはインターネットで使われる新しい通信形式です。

みんながインターネットでよく使ってるのは IPv4 です。

IPv6 は何が違うかっていうといろいろあるんですが、IPv4 との大きな違いはみんなにグローバルアドレスが割り当てられることです。グローバルアドレスっていうのは直接インターネットで通信できるアドレスですね。LAN 内のみのアドレスなんかはプライベートアドレスです。

グローバルアドレスくらい今でも貰ってるよ〜と思うかもしれませんが、普通のご家庭であればグローバルアドレスはせいぜい 1 個しか貰えない一方で、どこのご家庭でも PC3 〜 3 台はあるしスマホはあるしゲーム機もあるしで 5 台くらいはインターネットにつなぎたい機器があるでしょう。普通のご家庭でも 5 台くらいはね。それらにはローカルネットワークのアドレス (プライベートアドレス) が割り当てられ、インターネットとのやりとりはグローバルアドレスが割り当てられたルーターが仲介しています。

それに対して IPv6 の場合は全部の機器にグローバルアドレスを割り当てることができます。IPv6 だとおうちにグローバルアドレスが約 2 の 64 乗個分貰えます。さすがにそんなにいらないし全部は使えないんですけど、これなら PC が 20 〜 30 台あっても余裕で全部にグローバルアドレスが割り当てることができます。グローバルアドレスがみんなに割り当てられていると、家のルーターがインターネットとのやりとりを仲介する必要がなくなるのでちょっとすっきりします。まあいろんな都合で家のルーターは相変わらず必要なんですけど！

他に IPv4 との違いと言えば中身が全然違うことです。中身がどう違うかは気にしなくていいんですけど、全然違うものなので相互につながりません。IPv4 と IPv6 は (直接には) つながりません。つなげたい場合には途中で変換を入れる必要があります。

今まで IPv4 で足りてるのにそれとつながらない IPv6 を入れる必要ないじゃん……という気がするのはもっともですが、IPv4 もここまでインターネットが使われるとは思ってなかった時代にできたものなんで、一般ユーザーにはわかりづらい世界でいろいろと不都合が出ているようです。なんとか頑張って IPv4 を維持はしていますが、無理を重ねるより IPv6 に移行した方が素直ですよねということで、徐々に IPv6 への移行が行われています。ほんと徐々に。

ところで IPv4 と IPv6 は相互につながらないので、従来 IPv4 で見ていた Web サイトは IPv6 で

は見れなくなるのでしょうか？実のところ見れなくなるんですが、Web サーバーに IPv4 と IPv6 の両方でつながるようにしておくのは比較的簡単であり、現にそうしているサイトもありますので、IPv6 は従来のインターネットと別物になる！とかそういうわけではありません。とはいえ全部のサイトが IPv6 から見れるかというとそういうこともなく、むしろ 2019 年時点でもいまだに IPv6 で接続できないサイトも非常に多くあります。そのため IPv6 を使うときは IPv4 も同時に使って補完するのが普通です。徐々に IPv6 に移行はしていくはずですが、IPv6 のみで十分にインターネットを使えるようになるにはまだまだかかりそうです。

2. よくわからない「IPv6」

家で光回線を使ってると「IPv6」もしくは「IPoE」にするとインターネットが速くなる！という話を聞いたことがある人も多いでしょう。多いかどうかは知らんけど。

IPv6 は速いのでしょうか？いや実際はそんな速くないっす。IPv4 に比べて仕様が整理された IPv6 の方がネットワーク機器の処理が速くなる余地はありますが[1]、その差は家の回線での通信速度には大きく表れないでしょう。

じゃあ速くなるってのは嘘なのかというとそうでもないんですけど、IPv6 だから速いわけじゃないです。簡単に言うとフレッツ光の IPv4 は混んでるけど、IPv6 はなんとかなりそうなので速い (ことがある) てわけです。

日本の光回線だとフレッツ光[2]のシェアが高いのですが、フレッツ光の複雑な事情により IPv4 が遅くなってたり IPv6 が速いと言われることになっています。つまり NTT の光回線使ってない場合には関係ないです。他の会社でも IPv6 の方が速いことはあるかもしれないけど、まあそれはまたその回線業者の事情によるもので、一般的に言われてるのとは違う可能性があります。

フレッツ光は光回線ですが、直接インターネットにはつながっていない独自ネットワークを引いています。なにやら法律によって NTT 東西が直接インターネット接続を提供することができなくなっています。そのためみんながインターネットに接続する場合には、NTT 独自ネットワークからインターネットに出ていく装置を経由して接続しています。PPPoE というプロトコルでインターネットに出ていく装置に接続しているのですが、近年ではこのインターネットに出ていく装置が混んでいるので回線速度が遅くなっています。じゃあ装置を増強すればいいんですけど、この装置は NTT 東西が増強することになっており、どうもあんまり増やしたくないようです (お金もかかるし、IPv6 に移行してほしい？)。

ところでフレッツ光の独自ネットワークは IPv6 で構築されています。せっかくなのでこのままインターネットにつながれば便利なんですけど、NTT 東西は直接インターネット接続サービスを

1　実際速くなるかどうかは機器の実装によりますし、IPv4 には長年の最適化があるので絶対 IPv6 の方が速いとは言えないようです。

2　ここで言うフレッツ光は正確にはフレッツ光ネクストで、B フレッツはまた話が別なんですが、めんどうなのでフレッツ光と略します。B フレッツはもうほとんど無いので。

CS:GO サーバー改造事始め

著 : ぷろぐれ

Counter-Strike シリーズは 2019 年でリリースから 20 年を迎えたチーム対戦型 FPS です。Counter-Strike シリーズは mod (改造) がとても盛んです。単にゲーム中の演出を強化するものから、全く別のゲームにするものまで多種多様です。例えば「キルを重ねたときに Killing Spree の効果音と表示をする」、「感染によって増えるゾンビから逃げ続ける (ゾンビエスケープ)」といったものは mod をサーバーに導入する事で実現されます (但しゾンビエスケープなどはマップに組み込まれた複雑なスクリプトとの組み合わせで実現されます。マップに組み込むスクリプトはここでは扱いません)。 ここでは Counter-Strike シリーズの最新作である Counter-Strike: Global Offensive (以下 CS:GO) での mod サーバーの作り方を紹介します。ちなみに CS:GO は 2018 年 12 月に無料化しているので安心して読み進めてください。

CS:GO は Source Engine というゲームエンジンで作られています。 Source Engine は、1998 年に開発された GoldSrc エンジンの後継として 2004 年に Half-Life 2 に初めて採用され、以降バージョンアップを重ねながら様々なゲームに採用されています。 2018 年にリリースされた Apex Legends も Source Engine が採用されています (ただし 2019 年 9 月現在はカスタムサーバーを建てることができないので当記事の技術を使うことはできません。残念)。 CS:GO は 2012 年に最初のバージョンがリリースされました。CS:GO では当時の Source Engine のバージョンをベースとして独自のカスタマイズを含めていますが、ベースはどのゲームも同じです。 GoldSrc エンジンの頃から mod の開発は活発で、Source Engine でもその流れを引き継いでいます。 Source Engine には addon という mod のための仕組みが最初から用意されています。 addon はローレベルの仕組みなので、ほとんどの mod は addon を直接実装せず Metamod:Source と SourceMod を用いて、その plugin として実装されています。 Metamod:Source は addon を直接実装するときのいくつかの問題を解決する addon です。 SourceMod は SourcePawn という独自の言語で実装した plugin を読み込むことで、比較的簡単にサーバーを改造することができる addon です。 SourceMod は Metamod:Source に依存しています。 addon を直接実装しないための別の addon として Source.Python があります。 Source.Python も SourceMod と同様に追加で plugin を読み込んでサーバーを改造することができる addon です。 Source.Python は python(3.6) で mod を実装できるので、大規模な mod を実装する場合は表現力の乏しい SourcePawn を用いる SourceMod よりも遥かに楽に実装ができます。

1. サーバー構築

CS:GO は過去のシリーズと違い公式のサーバーが用意されていますが、過去のシリーズと同様に自前でサーバーを建てることもできます。 mod サーバーを作るには自前のサーバーを用意する必要があります。以降、環境は Amazon Linux2 または Ubuntu (18.04 LTS) を想定し、基本的な

Linux 知識がある前提で書かれています。

サーバーのインストール作業は以下の通りです。 /opt/steam/csgo-ds/ 以下に CS:GO をインストールします。

```
$ WORK_DIR=/opt/steam

$ sudo useradd -m steam                                  # サーバー用ユーザー作成
$ sudo mkdir --parents $WORK_DIR                         # インストール場所作成
$ sudo chown steam $WORK_DIR

### 依存ライブラリのインストール
### ↓ Amazon Linux2 の場合
$ sudo yum install -y glibc.i686 libstdc++.i686
$ sudo yum install -y zlib.i686 libffi.i686        # source.python
$ sudo yum install -y python3
### ↓ Ubuntu の場合
$ sudo apt install -y lib32gcc1 lib32stdc++6
$ sudo apt install -y unzip
$ sudo apt install -y python3-distutils
$ sudo curl -kL https://bootstrap.pypa.io/get-pip.py | sudo python3
$ sudo dpkg --add-architecture i386
$ sudo apt-get install -y zlib1g:i386 libffi6:i386 # source.python

### CUI 版 Steam のインストール
$ cd $WORK_DIR
$ curl -sqL "https://steamcdn-a.akamaihd.net/client/installer/steamcmd_linux.
tar.gz" | tar zxvf -

### Steam で CS:GO をインストール
$ ./steamcmd.sh +login anonymous +force_install_dir ./csgo-ds +app_update 740
+quit

### ↓ Ubuntu で steamclient.so のエラーが起きる場合に実行してください
$ mkdir -p /home/steam/.steam/sdk32/
$ ln -s $WORK_DIR/linux32/steamclient.so /home/steam/.steam/sdk32/
```

ここまででひとまず標準サーバーのインストールは完了です。

続いて以下の場所にファイルを作成します。

/opt/steam/csgo-ds/csgo/cfg/server.cfg

```
hostname サーバー名
sv_lan 0 // LAN内に限定する場合は1にする
sv_region 4 // Asia
```

そしてサーバーの起動のために、Steam のサイトから以下の 2 つのトークンを発行してください。

- **AUTH_API_KEY**

Steam コミュニティ :: Steam Web API キー : https://steamcommunity.com/dev/apikey

- **LOGIN_TOKEN**
  Steam コミュニティ :: Steam ゲームサーバーのアカウント管理 : https://steamcommunity.com/dev/managegameservers

トークンを発行したら、以下のコマンドでサーバーが起動します。それぞれのトークンは発行されたものに置き換えてください。尚、ファイヤーウォールが有効な場合は UDP の 27015 番からのアクセスを許可しておいてください。

```
$ WORK_DIR=/opt/steam

$ LD_LIBRARY_PATH=$WORK_DIR/csgo-ds:$WORK_DIR/csgo-ds/bin \
  $WORK_DIR/csgo-ds/srcds_linux -game csgo -usercon \
  -authkey AUTH_API_KEY \
  +sv_setsteamaccount LOGIN_TOKEN \
  +map de_dust
```

サーバーを起動したら、ゲームを起動しモードを「OFFICIAL MATCHMAKING」から「Community Server Browser」に変更して右下の「サーバーの追加」から IP を入力すると、あなたのサーバーが表示されるはずです。

Source Engine にはゲームやサーバーの振る舞いを変える convar (設定値) が大量に公開されています。これでもってサーバー名などの基本的な設定や、あるいはこれをいじるだけでも変なサーバーを作ることができます。 例えばサーバーのコンソールで以下の通りに入力すると、ゲーム内の重力を変更できます。

```
sv_gravity 200
```

ゲームに戻ると重力がものすごく軽くなったでしょう。こういった convar は server.cfg に記述しておくことでも効果を発揮します。 convar の一覧は Valve の開発者サイトで公開されています。ただし一部情報が古いので有志がまとめたものを参照するのも良いでしょう。「`csgo-list-of-cvars`」で検索すると出てきます。

2. addon の導入

引き続き、addon をインストールしていきます。以下では記述時の最新版の URL を記載していますが、各 addon のサイトで最新版を確認してください。

- Metamod:Source : https://www.sourcemm.net/
- SourceMod : https://www.sourcemod.net/
- Source-Python : https://forums.sourcepython.com/

VRC 民に効く (かもしれない) Unity 操作 Tips まとめ

著 : ぬか

VRChat を始めて Unity を触り始めた方が多いと思う。そこで基本的操作から、VRchat のアバターやワールド制作等にも関連しそうなもう一歩くらい進んだ Unity の操作をまとめる。

1. カメラ

- **右クリック +WASD**
  カメラ移動　QE で上下 さらに Shift で高速移動
- **右クリック＋ Alt ＋ドラッグ**
  カメラをゆっくりとズーム
- **Alt ＋ドラッグ**
  対象を中心にカメラを回せる
- **Shift ＋ Ctrl ＋ F**
  対象を現在の Scene のカメラの向きにセット（※画像 1）
  Camera に使うとシーンビューで見ている向きの通りに配置される。

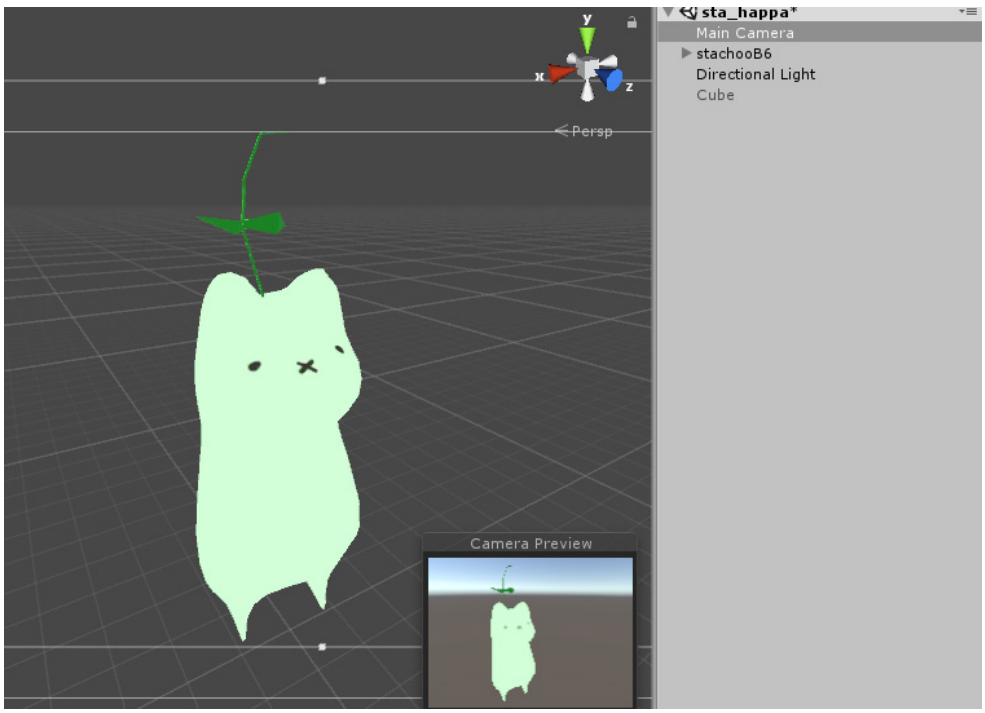

図 1. 画像 1

2. エディタ

- **Hierarchy やインスペクターのタブを増やす**

  メニューから Add Tab でタブをつくり、エディタ上の好きな場所にセット。鍵のアイコンでロックも可。(※画像 2) ロックしないと互いのタブに干渉してしまうので活用していきたい。

  タブを切り離すとマルチディスプレイ等にも効率的。

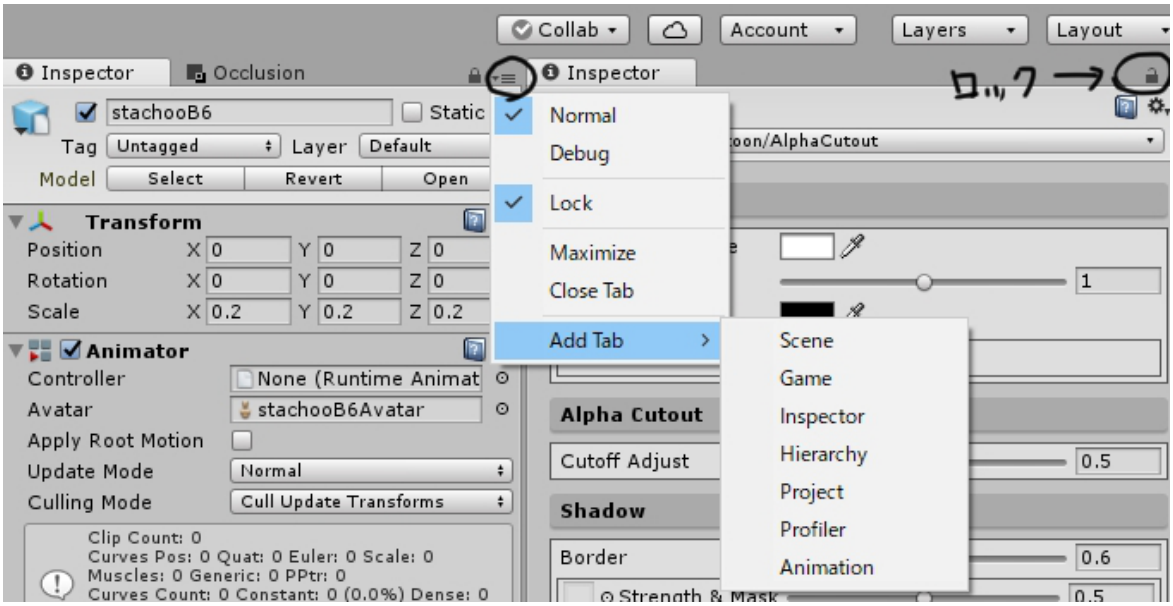

図 2. 画像 2

- **Hierarchy の検索欄**

  ファイルの種類やコンポーネント名を入力でアタッチされたオブジェクトを検索してくれる。さらに☆印をクリックで Favorites 入りし、検索状態を保存できる。ここの検索欄の消し残しで Hierarchy にあるはずのものが消えて一瞬パニクったりするのあるある。

- **オブジェクトの階層を全表示**

  Alt ＋クリック、階層構造を全展開してくれる。アーマチュアとかにどうぞ。(※画像 3)

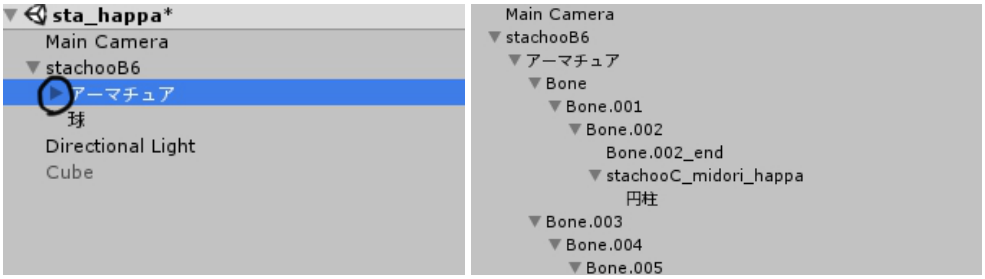

図 3. 画像 3

Misreading Chat リスニングガイド

著 :vivivi

1. 序文

「Misreading Chat」という CS（コンピュータサイエンス）の論文を読んで解説するポッドキャストが面白いので紹介します。

https://misreading.chat/ Misreading Chat - CS の論文読んで話をしよう

2018 年 3 月から途中何度か休暇も挟み、現在も週 1 ペースで更新されています。

技術系ポッドキャストは時事ネタや雑談系が多いですが、このポッドキャストでは論文を噛み砕くことに主眼を置いています。もちろん全部聴くのがおすすめなのですが、回数も多くなってきたので気になったものを分類しメモ書きをしてみようと思います。

2. 機械学習

#1a Tensor Comprehensions

Facebook が出した TensorComprehensions というフレームワークに依存しない機械学習の高性能な抽象についての説明です。お前は何を言ってるんだ、って感じですがいきなり初回でついていけるか不安になる題目です。簡単に言うと機械学習は単純な計算を膨大にやる事が多いのでそれを簡単に書けるようにしたり速くしたりってことらしいです。こういうのは実際自分で書いてみないとよくわからないですね。

#8 Generative Adversarial Nets

いわゆる GANs と呼ばれてるやつの説明です。ジェネレータとディスクリミネータを使いながら徐々に賢く画像を生成していくアレです。ジェネレータがディスクリミネータを騙せるような画像を生成できれば勝ちです。しかし機械学習はなぜそうなるのかがわからんがうまくいくから OK!、というものが多くてむず痒いですね。

#9 Automatic Differentiation in Machine Learning: a Survey

自動微分の論文です。ってなんやねんてのは説明してくれます。ようするに適当な数値から始めて計算してその傾きから少しずつそれらしい値に近づけていくものです。この回はサーベイ（調

査）論文で、機械学習の説明などの話になるのでかなりわかりやすい回になってます。

#15 Neural Machine Translation by Jointly Learning to Align and Translate

Attention を使って機械翻訳する論文回です。#15,51,53 で自然言語処理機械学習 3 部作になってます。Misreading 屈指の難解回で RNN や Attention を知らなければ全くわからないと思います。RNN を事前に予習してから聞くことをお勧めします。しかし機械学習はこういった難しいことをわからなくても使えてしまうのが面白いとも言えますね。

#16 A Deep Learning Approach for Generalized Speech Animation

ディズニー映画の口パクの部分を機械学習で作るという論文です。昔からこういうことやってるのかと思ったら意外に新しい論文なのでまだ新しい分野なのでしょう。バリバリ機械学習だぜ、っという論文ではないので聞きやすくオススメです。

#46 An Introduction to Neural Information Retrieval

機械学習で検索をやる入門の論文です。検索文字列から検索対象の文字列をさがして持ってくるというフェーズと、持ってきた検索対象をランク付けして出すというフェーズに分かれてるので、ランク付けのフェーズを機械学習でいい感じにしてあげると良い検索になるということです。ランク付けは昔からいろいろな手法があるので、機械学習しないランク付けから DL するランク付けまで説明してくれます。

#51 Attention Is All You Need

#15 で最後の方に出てきた Transformer の話です。Transformer とは RNN を過去のものにした Attention ベースの自然言語処理モデルです。中で森田さんが「全くわからなかった」というくらい難解でしたね・・・。当然難解回なので予習する、もしくは解説記事 (http://deeplearning.hatenablog.com/entry/transformer) を読む（これも難解だが）もしくは聞き流すのがよいでしょう。

#53 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Google が作った双方向 Transformer の説明です。超難解回。自然言語処理機械学習 3 部作の最

センテンス指向プログラミングの提案

著 :We

1. はじめに

マーチン・ファウラーさんの「流れるようなインタフェース」にインスピレーションを得て自分なりに考えた結果、やりたいことを文章で書いたら動く、と幸せなんじゃないだろうかというお話です。

サンプルコードはすべて C# で書かれています。

2. マーチン・ファウラーさんについて

あなたはマーチン・ファウラー (Martin Fowler) さんをご存知でしょうか。ThoughtWorks 社で働く技術者で、アジャイル開発やエクストリーム・プログラミングに精通したソフトウェア工学の第一人者です。著書も多数あり、『リファクタリング : プログラムの体質改善テクニック』などが有名です。昨年第２版が発売されましたが、まだ日本語版はありません。

マーチン・ファウラーさんはインターネット上でも精力的に活動しており、自らのサイトでさまざまなソフトウェア論を展開されています。

- Martin Fowler's Bliki
  https://martinfowler.com/bliki/

こちらはありがたいことに、有志による日本語訳サイトがあります。数百にもおよぶ記事が公開されています。

- Martin Fowler's Bliki (ja)
  https://bliki-ja.github.io/

3. 流れるようなインタフェース

前項の日本語訳サイトに、流れるようなインタフェースという記事があります。

- 流れるようなインタフェース
  http://bliki-ja.github.io/FluentInterface/

同記事では、流れるようなインターフェースがなんなのかについて、サンプルコードにて解説しています。時間の間隔を得るためのコードは、通常だと以下のようにしますが…

```
TimePoint fiveOClock, sixOClock;
...
TimeInterval meetingTime = new TimeInterval(fiveOClock, sixOClock);
```

流れるようなインターフェースではこうなります。

```
TimeInterval meetingTime = fiveOClock.until(sixOClock);
```

可読性が上がりましたね。より直感的になったとも言えます。ほかにもサンプルコードを用いて解説していますが、ここでは割愛します。流れるようなインターフェースについてさらに知りたい方は記事を参照してください。

もう一つ、例を出しましょう。文字列が null または空かを判断する場合、何も考えずに書くとこうなります。

```
string name = GetName();
if (name == null || name == "") return;
```

条件式に and や or があると一気に可読性が下がりますね。幸いにも、null または空の判定は専用のメソッドが用意されています。

```
if (string.IsNullOrEmpty(name)) return;
```

いくぶんマシになりました。さらに改良してみます。

```
if (name.IsNullOrEmpty()) return;
```

どうでしょうか。さらにすっきりしました。これは string に拡張メソッドを生やすことで実現できます。

```
public static class StringExtensions
{
    public static bool IsNullOrEmpty(this string value)
    {
        return string.IsNullOrEmpty(value);
    }
}
```

3.1. 可読性

なぜ可読性が上がったのでしょうか。文字数が減ったからでしょうか。それもあるかもしれませんが、私は手続き的な記載から文章になったからだと考えています。手続き的な記載の場合は、それが意味することを考える（変換する）必要がありますが、文章であれば読んでそのまま意味を理解できるからではないでしょうか。

最初のサンプルコードは、以下のような思考の経過をたどるかと思います。

著者紹介

kumaryu - IPv6 とつながらない P2P

ネットワーク全然わからんのに PeerCastStation という P2P 動画配信ソフトを作ってたりする人。

あとこの本を企画した。

ぷろぐれ - CS:GO サーバー改造事始め

普段は Web やスマホアプリのシステム作ったり面倒見たりする人。 FPS は Co-op しかやらん模様。札幌在住。

https://twitter.com/progremaster

ぬか - VRC 民に効く (かもしれない)Unity 操作 Tips まとめ

ゲームをつくったり絵をかいたりしています。

twitter　@nuka000

vivivi - Misreading Chat リスニングガイド

php や ruby などの LL でみなさんのめんどくさいことを自動化する仕事をしてます。

We - センテンス指向プログラミングの提案

Web とか .NET とか。今は iOS アプリ。勉強会もきまぐれにやってます。

https://nomipro.doorkeeper.jp/

twitter @RunLucky

めいまぁ - 表紙

pixiv129039

こんにちはこんばんは

peercast で絵を描く気力を出しています

配信出来なかったらやる気も出ません

2019
あけまして