スクリプト言語まとめ

山田 幸史朗，山崎 弘貴
Koshiro YAMADA , Hirotaka YAMAZAKI

1　はじめに

1980 年代から起こったインターネットブームによって，現在に至っても様々な Web アプリケーションが作られている．それらは全てプログラム言語で作られているが，システムの規模が大きくなり常時稼動しなければならないサイトやアプリケーションをコンパイラ方式の言語で作成するのは管理の面で難しくなっている．このような背景からコンパイルをしなくてもよいスクリプト言語が必要とされている．

本報告では，スクリプト言語についての起源とこれからの Web アプリケーションやシステムにおけるスクリプト言語の未来について語っていく．

2　スクリプト言語

1980 年代まで，プログラム言語は C 言語や Java のようにプログラマが書いたソースコードを，コンパイラという翻訳機によってマシン語（コンピュータにわかる命令）に変換し実行する，という動作を行うコンパイラ型方式が多数であった．なぜなら，コンピュータの利用方法が，プログラム実行を予約するバッチ型処理で，対話的処理が実用的ではなかったという背景がある．しかし，タイム・シェアリング・システム（TSS）によって，対話的処理が可能となり，ソースコードを一行ずつ逐一解釈し実行するインタプリタ方式の言語が登場するようになった．この方式が一般的になるに従って，設計思想や用途によって，アプリケーション組み込みの小規模な言語や簡単な制御構造を持つスクリプト言語が広まってきた．1990 年代になると，インターネットがビジネスの宣伝の広告塔となってきたため，静的な動作を行うのではなく，動的な Web サーバを作る Web プログラミングの必要性が出てきたため，よりスクリプト言語の実用性が深まった．現在では，24 時間止めることのできない Web サーバを扱うことも多くなった為，コンパイラ方式ではなくインタプリタ方式を採用したスクリプト言語の活動の場となっている．

スクリプト言語の特徴としては以下の点が挙げられる．

- メンテナンスが簡単
  コンパイルを必要としない為，システムの再起動を行う必要がなくいつでも変更が可能である．
- 習得が容易
  一般のプログラミング言語に比べて機能は少ないものの，記述がわかりやすく，誰にでもすぐに扱うことができる．
- 実行速度が遅い
  一行毎に構文解析を行いながら実行していくインタプリタ方式であるため，既に構文解析や機械語への翻訳が終わっているコンパイラ方式と比較するとどうしても実行速度が遅くなってしまう．

3　スクリプト言語の種類

スクリプト言語に共通していることは，必要な仕事を簡潔に記述し実行できることである．必要な仕事の例としてテキスト処理やグラフィックユーザインターフェース（GUI）などが挙げられる．以下に Perl，PHP のスクリプト言語の利点と特徴を述べる．

3.1　Perl

Perl は，1987 年にラリー・ウォールによって作られたスクリプト言語で，正式名称が「Practical Extraction and Report Language」と呼ばれることから，「様々なデータを処理してレポートを出力する事」を目的としている．アプリケーションの一部で稼動する従来までのスクリプト言語と異なり，OS のシステムコールを呼び出せるなど，C や Java に劣らない機能性を持っており，大規模なプログラムへの実装にも用いられる．また，Perl が出た当初においてテキスト処理を得意とし，様々なデータベースを利用できるという世間に求められていた二種を両方とも満たしていたため，爆発的な人気を誇り，CGI には Perl が用いられるようになった．CGI については，3.1.1 節において詳しく説明する．

3.1.1　CGI

CGI は，Web サーバが Web ブラウザからの要求に応じて，プログラムを起動するための仕組みである．（Fig.1 参照）従来の Web サーバは蓄積してある文書をただ送出するだけであったが，CGI を用いることによってプログラムの処理結果に基づいて動的に文書を生成し，送出することができる．例えば，CGI + HTML の機能を使うと掲示板を作ることができる．その他にも，パスワード

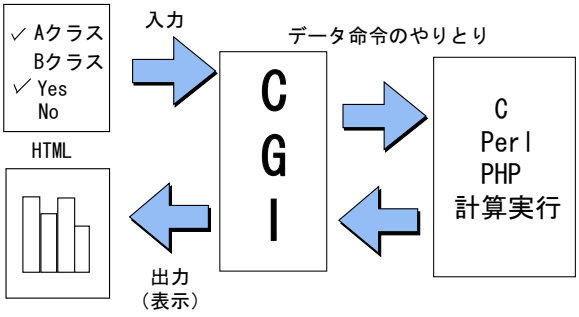


Fig.1　CGI の仕組み (出典：自作)

制限による会員制を行ったり，アクセス解析によって，どのような宣伝が効果的かを知ることができ Web 社会では必須の事項となっている．現在ではほとんどの Web サーバソフトが CGI に対応している．

CGI は，フォームからの入力や，プログラム言語の持つ機能も使うことができるが，以下のような事は行うことはできない．

1. サーバ側が任意でユーザ画面を書き換えるシステム
   CGI プログラムは接続状況の情報を持たないため，サーバ側がユーザ側のブラウザに働きかけることはできない．ユーザ側がリロードを行うことによって更新されるが，リアルタイムに情報を交換することはできない．
2. 別のサーバにあるファイルを読み書きするシステム
   CGI プログラムはサーバ内で動くプログラムであるため，同一サーバ内でない他のサーバのデータを読み込んで使用するといった手法はできない．
3. ブラウザの表現範囲を超える GUI
   CGI プログラムは結果をブラウザに返すため，表現がブラウザや HTML の枠から外に出ることはない．よって，グラフ表示に Excel を持ち込んだりすることはできない．

3.2 PHP

PHP は，1995 年にラスマス・ラードフによってモデルとなる PHP/FL が開発され，動的に HTML データを生成することによって，動的な Web ページを実現することを主な目的としている．この言語処理系自体は，C 言語で記述されている．Web サーバ上で動作し，Web サーバ上の文書が要求されるたびに，この文書に記述された PHP のプログラムを実行し，その結果を Web ブラウザに対して送信する．Web ブラウザに送信されるデータは通常の HTML であり，PHP のプログラムを含まない．PHP はサーバサイド側のプログラム言語であり，クライアントサイドと決定的に違うところである．この違いについては 3.2.1 節にて詳しく説明する．また，Perl との違いを Table1 に示す．

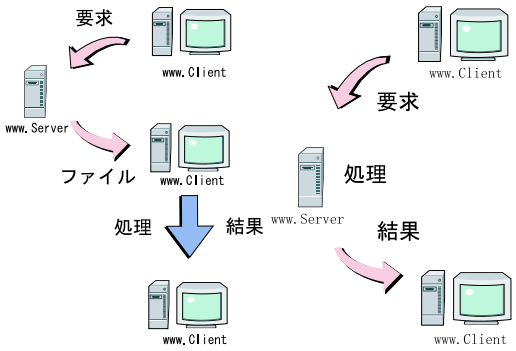
	Perl	PHP
実行位置	サーバ外部	サーバ内部
使いやすさ	変数が使いやすい	関数が使いやすい
DB の使用	難しい	容易
実行処理	html で返す	html に直接埋め込む
処理速度	遅い	速い
サーバの負荷	大きい	小さい

Table1　Perl と PHP の違い (出典：自作)

3.2.1 クライアントサイドとサーバサイド

クライアントサイドは，何万通りもあるコンピュータ上で動作するため，性能や OS，プラットフォームなども様々であり，全てクライアント側で動作するため，比較的大きなトラフィック量が必要となる．また，個人のコンピュータ上のデータを操作するため，変更が別のユーザに影響を与えることは無い．

一方，サーバサイド技術は基本的に動作するプログラムが管理されたサーバで動き，サーバの性能に合わせてチューニングでき，データベースとのやり取りは全てサーバ側で行われ，出来上がった結果だけを送るのでトラフィック量が少なくなる．また，データソースが全てのユーザで共有されるため，あるユーザが行った変更が別のユーザに影響することもあるため注意が必要である．Fig.2 に両者の違いと仕組みを図にしたものを示す．



(a) クライアントサイド　　(b) サーバサイド

Fig.2　クライアントサイドとサーバサイド (出典：自作)

3.2.2 今後の方向性

Java のコミュニティ JCP (Java Community Process) で，JSR-223 という Java の機能拡張の提案がある．Java で開発する際に，フロントエンドのページを作る部分に PHP を使える試みである．これにより，Java か PHP かという二者択一ではなく，ページの部分は書き慣れた PHP を使い，本格的にビジネスロジックを構築する部分は Java にするような柔軟な使い分けも可能となる見込みである．

4 オブジェクト指向スクリプト言語

現在の開発では，ソフトウェアの構造が従来のような 1 から開発するスクラッチ開発ではなく，データベースへの接続やアクセスや，API の接続機能を持つ等のソフトウェア部分を自分のプログラムと組み合わせて作るフレームワークが使われるようになっている．これは，アプリケーション又はシステムの規模が大きくなり，コストと時間が膨大になるため，頻繁に使用される部分を流用することで，開発効率の向上を行うためである．それに伴い，ソフトウェアの構造が非常に複雑になった為，人が理解しやすい単位が必要となる．それがオブジェクトという単位であり，オブジェクト指向を利用することで，ソフトウェア構造を分かりやすくすることができる．この方式が進展した為，オブジェクト指向を採用したスクリプト言語 Ruby が登場した．

4.1 Ruby

Ruby は，1995 年にまつもとゆきひろによって作られたスクリプト言語である．Smalltalk や C++などの本格的なオブジェクト言語ではなく，手軽なオブジェクト指

向プログラミングを実現するための言語である．Perl と同程度の処理能力やシンプルな文法に加え，例外処理やイレテータなどの制御構造のオブジェクト化により，非常に簡単なコードを書くことができる．

4.2 Ruby on Rails

Ruby on Rails はデータベースを利用した Web アプリケーションを構築するためのフレームワークで，主に Ruby と MySQL で構成される．MVC アーキテクチャを用いることで，開発作業を分担することができ，仕様変更の影響を少なくすることもできる．(Fig.3 参照)

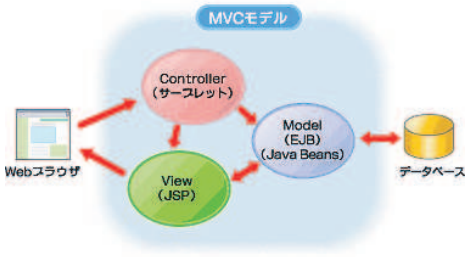


Fig.3 MVC の仕組み（参考文献 2 より引用）

また，新規に記述するコードの量が少なくて済み，簡単に Web アプリケーションを開発できることが特徴である．その為，データベースアプリケーションには向いているが，メールフォームやアンケートなど，遷移の少ないものではその利点をあまり発揮することができない．また PHP などに比べて，メモリーを消費すると言う問題もあり，資源の少ないサーバーでの運用には注意が必要である．Table2 に Ruby on Rails の特徴を示す．[3)]

名称	概要
Convention over Configuration	規約に則ることで様々な省力化が可能となる
Don't Repeat Yourself	同じデータやロジックを重複させない
環境タイプ	開発・テスト・本番の 3 環境が準備されており各工程に最適な特性を持っている
ジェネレーター	ソースコードの雛形生成ができる
フルスタック	画面周り，データ周りや自動テストなど開発に必要な機能がオールインワンで入っている

Table2 Ruby on Rails の特徴 (出典：自作)

4.3 今後の Ruby

Ruby は優れた設計思想やスクリプト言語におけるオブジェクト指向の実現と Web アプリケーション開発システムの Ruby on Rails によってスクリプト言語として世界的に用いられるようになった．それに伴い，Ruby の弱点として実行速度が指摘されることが増えてきている．そこで Ruby 2.0 が提案され，Ruby のままであり続ける事をポリシーに，Ruby を再デザイン，再実装し，不安要素をなくしたものであるとしている．しかし，その提案から既に 10 年が経過し，このままでは実装されるのが 2010 年になるのかもっと先になるのか分からない状態なため，2007 年のクリスマスに Ruby1.9.1 を発表を行った．この実装内容には，Ruby インタプリタの世界最速を目指し，2006 年 12 月 31 日に Ruby リポジトリにマージされ公式な処理系となった YARV[4)] のマージも含まれている．YARV は，笹田耕一氏によって開発されている Ruby 仮想マシン実装のひとつで，2004 年度に未踏ソフトウエア創造事業に採択された．現在の Ruby で用いられているインタプリタ方式の中間コードである，再帰によってノードツリーを辿る構文木型から，バイトコード方式のコンパイルを実行することで高速化を計っている．(Fig.4 参照) また，Ruby1.9.1 では，YARV のマージの他にネイティブスレッドへの対応やマルチ仮想マシン化が採択されている．

また，大規模なプログラムの実行時間が占める割合の多くである GC（ガベージコレクション）の短縮も考えられている．GC の高速化の手法として，仮想メモリのダーティビット情報を利用して，古い世代のオブジェクトから若い世代へのオブジェクトへの参照を検出することで GC の中断時間短縮，参照の検出のために必要なプログラムの修正コスト削減，参照の検出のオーバーヘッド削減を行うことで GC の総合的な短縮を計ろうとしている．[5)]

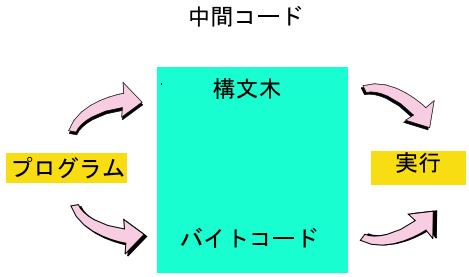


Fig.4 インタプリタ方式 (出典：自作)

5 まとめ

本報告は，Web の発達に伴い登場し，スクリプト言語に大きな変動を与えた Perl から PHP，そしてオブジェクト指向を取り入れた Ruby について述べた．今後，Ruby の進化と VM によって，大規模なシステムにも Ruby on Rails の使用が増えていくと考えられている．

参考文献

1) Rails が注目されている理由
http://www.thinkit.co.jp/free/article/0605/2/1/
2) MVC モデル
http://www.atmarkit.co.jp/fjava/javafaq/j2ee/j2e07.html
3) 未踏ソフトウェア創造事業
http://journal.mycom.co.jp/articles/2006/02/28/yarv/
4) Ruby2.0 新仮想マシン YARV
http://journal.mycom.co.jp/articles/2006/02/28/yarv/
5) RubyVM の設計と実装
http://lc.linux.or.jp/lc2001/papers/ruby-vm-paper.pdf