Column

アプリケーション開発とOSS

マイクロサービス時代におけるアプリケーション・サーバー、フレームワークの選択

本稿ではWebアプリケーション開発でよく使用されるオープンソースソフトウェア（以下、OSS）について、その概要や特徴をご紹介します。この領域で利用されるOSSはあまりにも多岐にわたるため、アプリケーション・サーバーと、ブラウザーにおけるレンダリングで用いられるOSSに限定して紹介します。1つのテクノロジーを選択してそこに多くのアプリケーションをデプロイするというこれまでの手法に加え、近年はマイクロサービス化により、サービスごとにその特性に合ったテクノロジーを選択するケースが増えてきています。そのため今後は、さまざまなテクノロジーの特性を把握した上で、その中から適宜、比較検討するケースが増えていくでしょう。

1. アプリケーション・サーバー

Webアプリケーション開発では、http通信などの基本的な機能をアプリケーション・サーバーに任せることで開発者は業務機能の開発に専念できます。

■ Open Liberty

2017年9月に発表されたOSS化されたWebSphere Libertyです。JavaEE（現在は、JakartaEEに改名[1]）に準拠しており、コンポーネントを分けて必要な機能のみで構成できるようにすることで、特にクラウドネイティブ環境で軽量なサービスを実装することを目標としています。Dockerイメージも用意されており、コンテナ環境でもすぐに使用することが可能です[2]。

■ Spring Boot

Springは、Rod Johnson氏が2002年10月にリリースした、Javaによるサーバー・サイド・アプリケーション開発のためのフレームワークです。当時の重厚長大なJavaEEアプリケーション・サーバーへのアンチテーゼとして、強力なDIコンテナとAOP（アスペクト指向プログラミング）をベースとして、その上にデータ・アクセスやトランザクション管理のためのフレームワークを簡単に構成できるようになっています。JavaEEにDIコンテナとしてCDIが標準仕様として導入され役目を終えるかにも思えましたが、現在も人気は全く衰えず活発に開発が続けられています。

2014年にリリースされたSpring Bootは、Tomcatを内蔵しており、すぐに動くアプリケーションを構築できることが特徴となっています。プラグインが整備されたEclipseをSpring Tool Suiteとして提供しており、開発者は面倒なセットアップをしなくても、すぐに動くサーバーを開発、構築できるという利点があります。

■ Node.js

Node.jsは、JavaScriptを用いてサーバー・サイドのアプリケーションを構築するためのフレームワークで、

Ryan Dahl氏によって2009年に開発されました。Node.jsを使用するとクライアント・サイド（ブラウザー側）もサーバー・サイドもJavaScriptで実装できるというのが利点の一つですが、Node.jsの特徴で有名なのはC10k problem（1万台クライアント問題）対応でしょう。Node.js登場当時の多くのアプリケーション・サーバーは、多量のクライアントを“同時に”さばけないという課題がありました。それは例えばSocket資源、スレッド資源、サーバー・サイドのセッション資源の管理が、多量のクライアントを同時に相手にするユースケースにマッチしていなかったことが原因の一つです。Node.jsはこうした問題を考慮して設計されているため、多量のクライアントをさばかなければならない場面に向いています。ただし、このためにはノン・ブロッキング・プログラミングモデルで開発しなければならないため、開発の難易度は高く、また障害時にコードの流れを追うのが面倒になるケースもあり、開発チームが十分に使いこなせるかどうかを検討してから採用した方が良いでしょう。

■ Playframework

Javaによるサーバー・サイド・アプリケーション構築のデファクトはJavaEEですが、非JavaEEのフレームワークも幾つか存在します。Playframeworkは2009年に1.0がリリースされ、現在は2.8が最新です。プログラミング言語としてJava以外にScalaが利用可能で、この後解説するRuby on Railsのように、多くの機能（データ・アクセスからテストスイートまで）をパッケージングして簡単に開発できるのが特徴です。フロントエンドにNettyを用い、データアクセスもノン・ブロッキングのコンポーネントを選ぶことが可能なので、C10k problemにも対応できます。

■ Django

Djangoは、Pythonで実装されたWebアプリケーション・フレームワークです。その歴史は古く、最初のリリースは2005年7月に遡ります。Djangoは素早い開発に注力しており、特に典型的なCRUDアプリケーションは、ほとんどプログラミング無しで開発が可能であり、適切な適用先を選べば大きな開発生産性を得ることができるでしょう。ただしPythonの実行性能はあまり高くはないので、適用にあたっては事前にパフォーマンス検証をしておくと良いでしょう。

■ Ruby on Rails

Ruby on Railsは、2004年7月に公開されたRubyで実装されたWebアプリケーション・フレームワークです。スキャフォルドと呼ばれる仕組みで簡単にCRUDアプリケーションの雛形を生成できる点が脚光を浴びました。しかし、Ruby on Railsの本質はむしろ、徹底した自動テストのための仕組みの提供、データベースの設計をアプリケーション・コードと同期して管理できるマイグレーション、Webコンテンツ・キャッシュを効率良く実装するためのアセット・パイプラインや、リクエストをまたいで情報を伝達できるflashなど当時のWebアプリケーションが弱かった部分に光を当てて、簡単に開発できるフレームワークとしてまとめ上げた点にあると言えるでしょう。当時大きなブームを巻き起こし、またその後に誕生したWebアプリケーション・フレームワークにも多大な影響を与えました。Rubyも実行性能はあまり高くないので、適用にあたっては事前にパフォーマンス検証をしておくと良いでしょう。

2. ビュー・レンダリング

Webアプリケーションでは、最終的にHTML(DOM)をブラウザー上にレンダリングしますが、これには大きく分けて2つの方法があります。1つ目はサーバー上でHTMLを生成してブラウザー側に返送してブラウザーがレンダリングする「サーバー・サイド・レンダリング」、2つ目はブラウザー上で稼働するJavaScriptのコードがサーバーと通信して、その結果を基にブラウザー上のDOMを更新する「クライアント・サイド・レンダリング」です。クライアント・サイド・レンダリングで実現された、1つのページで動作するアプリケーションのことを、SPA(Single Page Application)と呼ぶこともあります。前者は制御が単純で開発が容易であり、またブラウザー側に高度な機能が要求されないため、家電の組み込みブラウザーのような環境でのレンダリングもあまり苦労せずに行える反面、複雑なUXの実装が困難という欠点があります。逆に後者は複雑なUXを実現できる反面、クライアント側のコードが複雑化する欠点があります。

2-1. サーバー・サイド・レンダリング

サーバー・サイド・レンダリングの機能は歴史的に古いため、通常はアプリケーション・サーバー側で機能が提供されます。大別するとHTML内に特殊なタグを入れて、このタグによって生成されるHTMLを動的に生成する手法と、HTML内に特別な属性を入れて、レンダリング・エンジンがこれを利用してHTMLを動的に生成する手法があります。前者はHTMLとは異なる特殊なタグが必要となるため、デザイナーに敬遠される傾向があります。これを緩和するためのタグ・ライブラリーなどが利用された時代もありましたが、逆に複雑化してしまう欠点もあり、現在では下火になっています。

■ Java

JavaEEの仕様ではJSPが最も古くからあるサーバー・サイド・レンダリングのためのエンジンですが、現在の開発ではJSF(JavaServer Faces)を用いるケースが多いでしょう。JSFを用いることで画面上での部品の独立性が高まりビジュアル・エディターなどでの編集が容易になります。JSFもJavaEEの仕様に含まれるため、通常はアプリケーション・サーバーに含まれているものを使用するケースがほとんどでしょう。OSSの実装としてはApache MyFacesやPrimeFacesが挙げられます。

JavaEE標準以外のサーバー・サイド・レンダリングのテンプレート・エンジンとして他に主だったものとして、Apache Velocity、Apache FreeMarker、Thymeleafが挙げられます。前の2つはJSPと同様に特殊なタグをHTML内に入れることで動的なレンダリングを実現します。Thymeleafはこうした特殊なタグを用いるのではなく、HTML属性を入れることで動的なレンダリングを実現します。デザイン・ツールとの相性が良く、そのままブラウザーで開いてデザインを確認することもできるため、デザイナーの抵抗が少ないでしょう。

■ Java以外

Ruby on Railsでは標準でERB(Embedded Ruby)が提供されています。これはJSPと同様にRubyのコードをテンプレート内に埋め込むための仕組みです。Djangoも標準でテンプレート言語が提供され、特殊なタグを用いて動的要素を埋め込むことが可能です。

2-2. クライアント・サイド・レンダリング

初期のクライアント・サイド・レンダリングは、ブラウザー上のJavaScriptでXMLHttpRequest（Ajax）機能を用いてサーバーを呼び出した後、その結果に従ってブラウザー内のDOM要素を直接操作していました。特にこの時期はブラウザー間でのJavaScriptやDOM制御の方言が多く、これらを吸収して生産性を高めるツールとしてjQueryが広く用いられました。

クライアント・サイド・レンダリングは、ブラウザー上でレンダリングを行うため、ブラウザー上で実行するJavaScriptおよびCSSのコード管理が重要となります。特にブラウザー上に表示されるパーツごとになるべく独立して開発できるようにコンポーネント化したり、状態の遷移に対して無駄なくブラウザーのDOMを更新する必要があります。こうした機能を提供するのがSPA用フレームワークで、代表的なものとして、React、Vue.js、Angularが挙げられます。

■ React

ReactはFacebook社によって開発され、主にテクノロジーに強みを持つ企業に多く採用されています。仮想DOMを利用しており、アプリケーションからのDOM更新要求は仮想DOM上に実施された後、現在の実際のブラウザー上のDOMからの差分計算を行ってから、必要最低限の更新を計算して実行します。このため、特に複雑なページでは直接アプリケーションでDOMを更新するよりも高速にレンダリングすることが可能です。軽量なフレームワークで、JavaScriptの知識があれば、あまり学習コストをかけずに習得できます。

■ Vue.js

Vue.jsは、Angularの前身であるAngularJSの開発者の一人だったエヴァン・ヨー氏によって2014年2月にリリースされました。世界的に見ると現在はReactが優勢ですが、Vue.jsは日本や中国といったアジア圏で特に人気のあるフレームワークです。Vue.jsも仮想DOMの考え方を採用しており、また複数のコンポーネントにまたがる状態を管理するためのコンポーネントであるvuexの使い勝手が良い点が特徴です。状態管理が複雑な場合に用いると良いでしょう。

■ Angular

Angularは、その前身であるAngularJSが、2012年6月にリリースされ、その後2016年9月にバージョン2がAngularと改名されてリリースされました。AngularはReactやVue.jsとは異なり仮想DOMを用いていませんが、アルゴリズムを工夫することで大きなパフォーマンスの差が出ないように改善されてきています。3つの中では、最も手厚いフレームワーク（追加機能をインストールしなくても標準で多くの機能を用意）で、特に大規模開発が意識されたモジュールの概念が入っているのが特徴です。

[参考文献]
[1] Jakarta EE, https://jakarta.ee/
[2] Docker Hub : open-liberty, https://hub.docker.com/_/open-liberty

日本アイ・ビー・エム株式会社
グローバル・ビジネス・サービス事業　クラウド・アプリケーション開発
コンサルティング・ITスペシャリスト

花井 志生
Hanai Shisei

入社当時はC/C++を用いた組み込み機器（POS）用のアプリケーション開発に携わる。10年ほどでサーバー・サイドに移り、主にJavaを使用したWebアプリケーション開発に軸足を移す。2015年夏からクラウドを用いたソリューションのテクニカル・コンサル、PoCを生業としている。主な著書にJava、Ruby、C言語に関するものがある。