1.1 Google Gearsの概要

1.1.1 Google Gearsとは？

2007年5月31日、Googleは「Google Developers Day」というイベントを大々的に開催しました。「Google Developers Day」は、Googleにとって初の開発者向けイベントということもあり、世界10都市で同時に開催するという非常に華々しいものでした。そのイベントで何が起きるのか、世界中のWeb開発者が注目していたのです。

その場において、目玉技術の1つとして発表されたのが**Google Gears**です。Google Gearsの持つ、圧倒的に斬新なコンセプトは瞬時に大きな注目を集め、発表から数か月経った本書の執筆時点でも、世界中の開発者やWebサイトの提供者から熱い視線を注がれ続けています。

では、**Google Gears**（以下、Gearsと略します）のコンセプトとは一体なんなのでしょう。それは、「Webアプリケーションをオフラインでも動作させる」というものです。

「オフライン」という言葉をいきなり使いましたが、その意味がおわかりでしょうか。「オフライン」とはつまり、単純に「ネットワークに接続していない」ということです。今のところ、Webアプリケーションといえば当然インターネットに接続して利用するのが当たり前です。何せ、「Web」アプリケーションなのですから。しかし、Gearsが登場した今となっては違います。つまり、今まで「オンライン」で利用するのが当然と思われていたWebアプリケーションを、ネットワークから切り離された環境でも使えるようにする技術、それがGearsなのです。

今はまだ、Gearsを利用したサイトの数は多くはありませんが、この先様々なWebアプリケーションに採用されることは間違いないと思われます。Gearsが、Webテクノロジーをさらに一段階進化させるための重要な中核技術であることは疑いようもありません。

■ Google Gearsの正体

では、Gearsとは具体的にいうとどういう技術なのでしょうか。

Gearsの正体は、Webブラウザに対して追加する拡張機能（プラグイン）です。ユーザの視点から見ると、Gearsのインストールさえ行えば、オフライン状態でもWebアプリケーションを動かすことができるよう、ブラウザを強化することができます。しかし、Gearsを導入すればどんなWebアプリケーションでもオフラインで動かせるようになるわけではありません。Gearsが効果を発揮するのは、あくまでオフラインでも動作するよう作り込まれたWebサイトに対してのみです。

逆に開発者の視点から見ると、Gearsがインストールされたブラウザ上では、普段取り扱うことのできないJavaScriptのAPIを使用することができます。それらのAPIを使用すると、非常に容易にWebアプリケーションをオフラインに対応させることができます。Gearsが具体的にどういったAPIを提供しているのかについては、第2章をご覧ください。

GearsはWebサイトからフリーで入手することが可能です。バージョンに関していえば、本書執筆時点（2007年9月）では5/30発表時点のものから変化していません。しかし、開発者向けのプレビュー用として0.2.2というバージョンがリリースされています。開発者向けのバージョンではいくつか機能の追加、変更が行われており、本書もこのバージョンを元にして説明を行っています。

またGearsはオープンソースでもあります。採用されているライセンスはNew BSD License（修正BSDライセンス）という非常に制限の少ないライセンスです。そのため、開発者はソースコードを入手して、Gearsの内部構造を調べたり、Gearsそのものに修正を加えてビルドしなおすことも可能です。

Gearsは幅広いOSとブラウザをサポートしています（表参照）が、本書執筆時点では、残念ながらApple Safariブラウザには対応していません。

Google Gearsが対応しているOSとブラウザ

OS	ブラウザ	バージョン
Microsoft Windows XP/Vista	Internet Explorer	6以降
	Firefox	1.5以降
Apple Mac OS X 10.2 以上	Firefox	1.5以降
Linux	Firefox	1.5以降

■ Gearsが登場した背景

さてここで、Google Gearsがどういった流れで登場したのかをおさらいしてみましょう。

Gearsが登場した背景には、近年の**Webアプリケーションの高機能化**が挙げられます。数年前Webアプリケーションがまだ単純だった頃は、「ページの遷移」を基本としたユーザインターフェースと、複雑なサーバサイドのロジックの組み合わせでWebサイトが構築されていました。この頃は、Webページに求められる役割は単なるサーバへの「登録フォーム」でした。

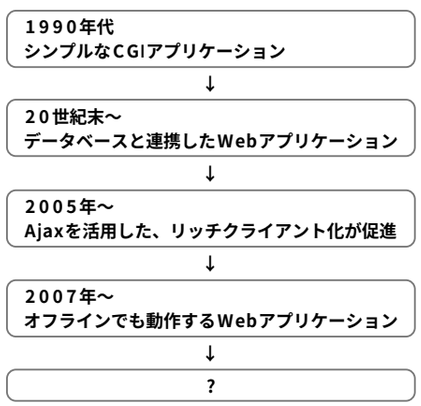
しかし、Ajax（Asynchronous JavaScript And XML）の登場で状況は一変します。Ajaxは、JavaScriptが非同期でサーバと通信するための単純な技術です。しかし何といっても、Ajaxがもたらした一番のメリットは「ページ遷移を発生させることなく、サーバにデータを送信できる」ことでした。これにより、「送信ボタンを押すとページ遷移が発生し、なかなかレスポンスが戻ってこないのが当然」といったこれまでの常識を打ち破り、デスクトップアプリケーションと遜色ないほどの使い勝手を持つWebア

プリケーションが多数生み出されるに至ったのです。

こうした流れは、「Webアプリケーションの使い勝手の悪さを改善する」といった意味合いを持っています。そして**次に改善すべき「使い勝手の悪さ」として注目されたのが、「オンライン状態でしかアプリケーションを使用できない」という点だった**のです。

「Webアプリはインターフェースの使い勝手が悪い」という常識はAjaxによって覆されました。そして、「Webアプリはネットワークにつながっていないと使えない」という常識は今覆されようとしています。こうした流れの中生み出されたのがGoogle Gearsである、と言うことができます。

Webアプリケーションの進化



オフラインWebアプリケーションというコンセプトが、Gearsによって突然もたらされたのではなく、こうした大きな流れの中で生み出されたという事実は意外と重要です。

まず、自分のWebサイトをオフラインに対応させたいと考えた場合、先ほど述べたようなWebアプリの進化の流れを踏襲する必要があるかもしれません。

Gearsは、ページ遷移をベースとした古いタイプのWebアプリケーションをオフライン化するのにはあまり向いていません。なぜなら、Gearsが想定しているアーキテクチャにそぐわないからです。Gearsは、「サーバサイドはサービス（API）を公開し、リッチなクライアントがそれを利用する」という、近年一般的になってきたWebアプリケーションの構成に最もうまくフィットします。そのため、現在Webで提供しているサービスをオフライン対応させたいと考えた場合、**Gearsを使うよりも先に、サイトのリッチクライアント化（Ajax対応）を行う必要が生じるかもしれません**。

さらに、「オフラインでも動くWebアプリケーション」というコンセプトは、Gearsによって突然もたらされたものではないことから、それを実現するための技術は他にも

存在します。

PDFやFlashで有名なアドビシステムズからは、「AIR（Adobe Integrated Runtime）」と呼ばれるプラットフォームが提供されています。大雑把に言うとAIRは、Webテクノロジーを用いてデスクトップアプリケーションを構築するための技術です。AIRでも、オフライン動作を実現するための独自APIが提供されています。

また、Zimbraはメーラやアドレス帳をブラウザ上で利用できるAjaxアプリケーションですが、独自のオフライン機構を用いたZimbra Desktop（http://www.zimbra.com/products/desktop.html）をリリースしています。

他にはWebブラウザであるFirefoxも、バージョン3においてオフラインWebアプリケーションを実現するための機能強化が実現される予定です。Gearsは、こうした様々なオフラインテクノロジーの中における、1つの（そして有力な）アプローチであるということができます。

1.1.2 Google Gearsの構成要素

ではさらに、Google Gearsの実体を深く掘り下げて見ていきたいと思います。

Gearsでは、オフラインWebアプリケーションを構築するために必要な機能がコンパクトにまとめられています。以下にあげる3つの構成要素がそれで、これらを効果的に用いることによってWebアプリをオフラインでも動作するようにできます。これらの構成要素について、開発者としてさらに詳しく学びたいという方は、第2章を参考にしてください。

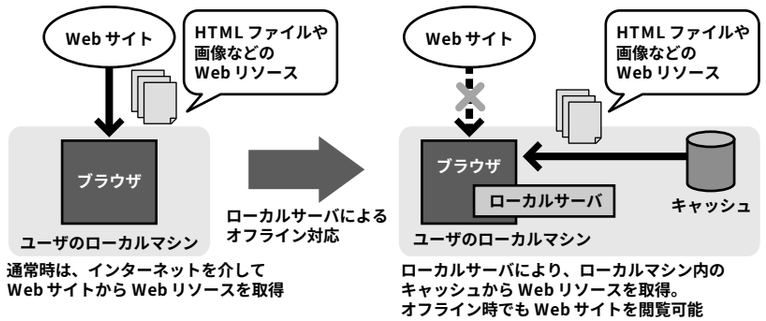
■ ローカルサーバ

まずオフラインアプリに求められる最低限の機能として、ネットワークに接続できない状態でもWebページを表示できる必要があります。この要求を満たすための機能が**ローカルサーバ**です。ローカルサーバとは、「サーバ」と名前が付いていますが、ネットワークのポートでリクエストを待ち受けるような一般的な意味のサーバではありません。

ローカルサーバの役目は、HTMLファイルや画像ファイルなどのWebリソースを、ブラウザがインターネットからダウンロードしようとする動作を横取りして、代わりにローカルにキャッシュ（保存）されたファイルを読み込ませることです。こうすることで、ブラウザはインターネットに接続できていない状態でもページを表示できます。

ローカルサーバによるWebリソースのキャッシュ方法には、2種類あります。1つは、**あらかじめ決めておいたリソースのみをキャッシュの対象とする**もの。こちらは、キャッシュの更新（最新状態への同期）が自動的に行われるのが利点です。もう1つは、**キャッシュするWebリソースを動的に決定するもの**です。こちらは、比較的自由にキャッシュするリソースを選べるのが利点ですが、キャッシュの管理は自分で行う必要があります。

ローカルサーバの動作

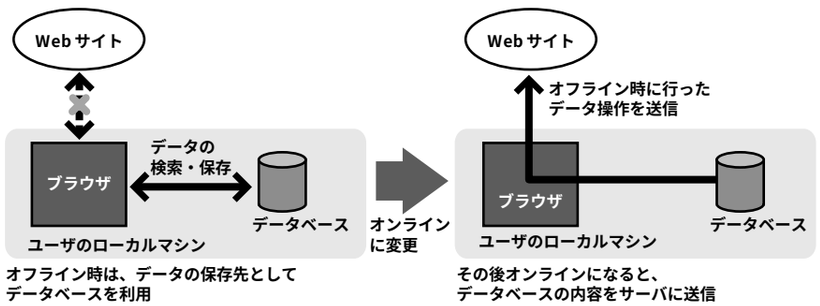


■ データベース

ローカルサーバを用いても、Web上のリソースをオフラインでも閲覧できるようになるだけです。本格的なオフラインWebアプリケーションでは、オンライン時と同様の操作、例えばデータの更新をオフラインでも行えるようにする必要があります。そうしたオフライン中にユーザが行った操作を一時的に記憶しておくために、**Gearsはリレーショナル・データベースを内包しています**。Gearsに組み込まれたデータベースは、SQLiteというオープンソースソフトをベースとしています。SQLiteはC言語で記述された非常に軽量なデータベースソフトですが、SQL（データベースの問い合わせやデータ更新に用いられる汎用的な言語）を用いた高度なデータ操作が可能で、パフォーマンスも良好です。

データベースという形式をとっているとはいえ、クライアントのローカルディスクに読み書きを行うという点ではセキュリティ上のリスクが生じます。Gearsはドメインごとにデータベースへのアクセス制限を課す事でセキュリティを確保しています。つまり、ドメインa.comが作成したデータベースに、ドメインb.comがアクセスすることはできません。

データベースを利用したオフライン動作



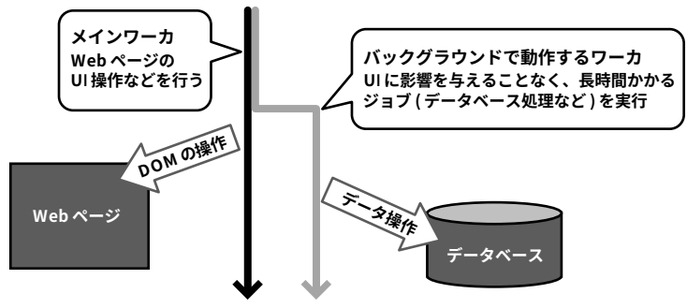
■ ワーカプール

ローカルサーバとデータベースを用いて、オフラインでも動作する高度なWebアプリケーションを作成できることがわかりました。それは同時に、アプリが持つ機能の多くをJavaScriptで記述する必要があるということを意味します。しかし、JavaScriptを用いて非常に時間のかかる処理―例えば数百～数千件のデータベース更新など―を行うと、その処理中はブラウザの描画処理自体が止まってしまい、アプリケーションが「フリーズ」したように見えてしまったり、ブラウザに処理を中断されたりすることがあります。以下の画像に示すようなダイアログに遭遇したことはありませんか？

時間のかかる処理をJavaScriptで行い、ブラウザに処理を中断されたことを示すダイアログ

こうした問題を解決するために、Gearsが提供している機能が**「ワーカプール」**です。「ワーカプール」は、長時間かかる処理をメインの処理とは別の「ワーカ（作業者）」に割り振り、バックグラウンドで実行させることができます。ワーカプールを用いれば、非常に時間のかかる処理をバックグラウンドで実行しながらも、ユーザは他の操作を行うことができるようになります。

ワーカを用いた仕事の分担



これらの機能を軸としつつ、Gearsは未だ進化の途上にあります。開発者やユーザのフィードバックを取り入れ、さらに高機能なオフラインアプリを容易に作成できるよう、着々とその機能を増やしつつあります。本書の執筆時点における最新バージョンは0.2.2ですが、それに至る過程でHttpRequestやTimerといったAPIが追加されました。こうした進化は恐らく当分の間続くことでしょう。