Webアプリケーションの応答性と開発効率を向上させるページ構築手法

ページ遷移に伴う無応答時間を短縮し，Webサーバの開発効率を向上

HTML5（Hypertext Markup Language 5）で策定された新しい通信プロトコルなどにより，Webサーバでのデータ変更を，即座に低負荷でブラウザに伝えることが可能になりました。監視制御システムなどで，刻々と変化するデータを可視化しながら，障害発生を一定時間以内に表示するようなアプリケーションのWeb化が現実的となりました。しかし，既存のWebアプリケーションフレームワークからの恩恵を受けにくく，開発効率の低下やアプリケーションの応答性に問題がありました。

そこで東芝は，ページ構築方法の最適化とブラウザからSQL（Structured Query Language）を発行できる仕組みの構築により，応答性と開発効率を向上させました。

Webアプリケーションの課題

Webサイトの表示は，**図1**に示すように，ブラウザからWebサーバにページ要求を送信（①），Webサーバがデータベース（DB）のデータを検索しページを構築（②），構築したページをブラウザに送信（③），ブラウザがページを表示（④），という手順で行われます。

この手順では，手順①を省略し，Webサーバ側のタイミングでページを送信してブラウザに表示させることはできません（**図2**）。これはブラウザとWebサーバ間の通信プロトコルであるHTTP（Hypertext Transfer Protocol）の制約によるものです。この制約のため，DB側のデータ変更に応じてブラウザの表示を変化させるには，ブラウザからページ要求を出し続ける必要がありました。

社会インフラシステムの監視制御システムでは，刻々と変化するデータを可視化し続けるとともに，障害発生を一定時間以内に表示できなければなりません。このため，Webサーバ側にページ要求を繰り返すことになり，Webサーバの負荷が増大します。

この問題に対処するため，HTTPを拡張したHTTPストリーミング(注1)（**図3**）と呼ばれるプロトコルや，より負荷の軽いWebSocket（**図4**）などの新しいプロトコルが標準化されています。

これら新しいプロトコルを使用する場合は，これまでのHTTPとの差異が大きいため，既存のWebアプリケーションフレームワークの恩恵を受けにくくなり，開発効率が低下するという問題があります。また，ページ遷移（ページ移動やページの部分的変更）が高頻度で発生するため，低頻度では問題とならなかった無応答時間（ページ初期化などに伴いユーザーへの応答ができなくなる時間）への対策が必要です。

ページ構築をブラウザ側に移植し無応答時間を短縮

ブラウザで動作するJavaScript(†)はシングルスレッドで逐次的に動作するため，処理中はユーザーに対し応答できません。無応答時間の削減には，個々のページ遷移に要する時間を減らす必要があります。そこで，**図5**に示すように，Webサーバで行っていたページ構築をブラウザで実行するように移植し，Webサーバはページ構築に必要なデータだけを送信するように変更しました。

この場合，ブラウザでのページ構築には時間を要しますが，表示中のページやデータを流用できます。それらを可能なかぎり再利用することで，個々のページ遷移における更新領域を狭めるとともに，多くの初期化処理を省略した結果，転送データ量を削減できました。これにより，処理全体としてのページ遷移時間は大幅に減少し，無応答時間も短縮できました。また，この方式ではWebサーバがデータ表示方法に依存しないため再利用性も高まりました。

ブラウザからのSQL発行を可能に

ページ構築をブラウザで行う場合，Webサーバはブラウザに API（Application Programming Interface）を提供する必要があります。このAPIは，Webアプリケーションの多様な要求に応える柔軟性を持ち，かつセキュリティ上の問題がないものでなければなりません。

監視制御システムの場合には，そのシステム要件から，多くの場合単純なDB問合せ言語のSQLをブラウザから発行できるAPIがあれば必要十分なことがわかりました。しかしブラウザからSQLを発行する場合，見せてはならないテーブルにアクセスされるなどのおそれがあり，セキュリティ上の問題が発生します。

そこで**図6**に示すように，WebサーバでSQLを解析し，不正なテーブルや，カラム，レコードなどにアクセスしていないことを保障するとともに，テーブル名やカラム名などのテーブル構造を隠蔽できる仕組みを導入しました。

これにより，簡単な設定だけでDB上のデータを，ブラウザからセキュアに取得したり変更したりすることが可能になりました。またSQLでは記述困難な処理については，SQLを拡張し対応しています。

今後の展望

今回の開発によって得た知見を元に，Webアプリケーション開発をより効率的に行うためのフレームワークを開発しています。より多様なアプリケーションのWeb化にも対応できるよう機能拡充を行うとともに，広く普及に努めていきます。

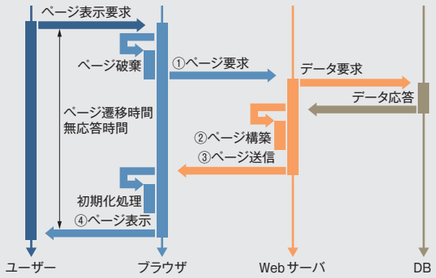


図1．Webサイト表示までの流れ — ブラウザからのページ要求を契機に，データ検索からページ構築までをWebサーバで行います。

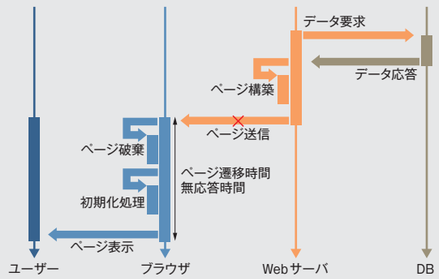


図2．実現できない処理の流れ — HTTPでは，ブラウザからのページ要求なしに，DB上のデータ変更に応じてWebサーバからブラウザにページを送信し，表示を変化させることはできません。

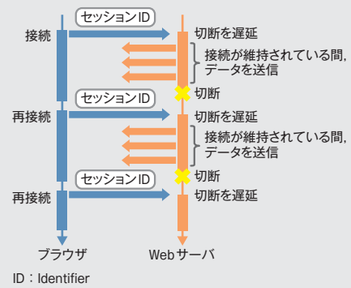


図3．HTTPストリーミングの仕組み — HTTPでの切断を可能なかぎり遅延させ，切断が発生するまでの間は，Webサーバから任意のタイミングでデータを送信できます。切断後はすぐに再接続し，接続状態を維持します。

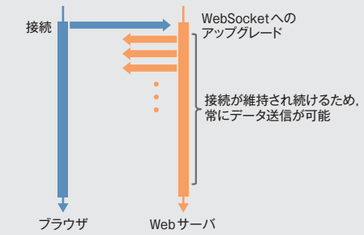


図4．WebSocketの仕組み — HTTPでの接続後，WebSocketプロトコルへアップグレードして接続を維持し続けます。接続が維持されるためデータ変更を即座に通知できます。また，セッションIDも不要です。

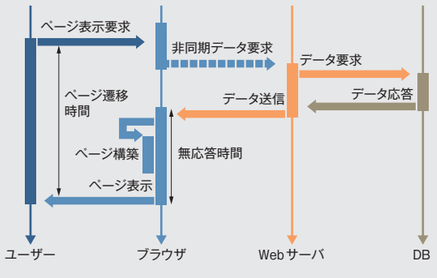
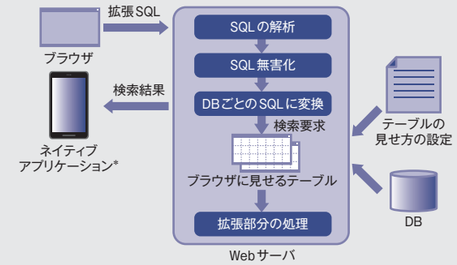


図5．ブラウザでのページ構築実行によるページ遷移時間の削減 — Webサーバで行われていたページ構築をブラウザ側で実行させ，既存のページやデータを活用して更新領域を狭め，ページ遷移時間を削減するとともに更新方法の柔軟性を向上させました。



*端末内で直接実行可能なアプリケーション

図6．ブラウザからのSQL発行 — DB上のテーブルに対して簡単にSQLを発行できる仕組みを構築しました。これにより，多くの部分は簡単な設定だけでWebサーバ側のアプリケーションを開発でき，開発効率が向上しました。

(注1)　投稿サービスのストリーミングAPIなどに使用。

古城　仁士
ソフトウェア技術センター
先端ソフトウェア開発担当