L_009

モバイル環境向けWebアプリケーションのセッション管理方式

Session Management for Web applications on Mobile Environment.

宗形 聡† 樋地 正浩†
Satoshi Munakata Masahiro Hiji

1．はじめに

従来 Web アプリケーションは，LAN に接続されたデスクトップ PC からの利用が中心であった．しかし近年では，ノート PC や携帯電話などのモバイル端末から無線 LAN，携帯電話網を通して Web アプリケーションを利用する機会が増加している．将来ユビキタス環境が実現されれば，ネットワーク機能を持つ組込機器でも Web アプリケーションの利用が増加すると考えられる．

Web アプリケーションのサーバとクライアントは，通常 HTTP で通信する．買い物サイトなどに代表されるように，クライアントは Web アプリケーションのサービスを利用する際，複数の Web ページにわたりリクエストを行う．一方，HTTP はステートレスなプロトコルのため，サーバ側でリクエストごとにクライアントの識別を行うことができない．そのため，サーバ側はセッション管理機能を持ち，クライアントの識別とセッション期間中の状態保持を行っている．セッションの有効期間は，サーバのメモリリソース解放や，セキュリティ向上のために数分間と短く設定されることが多い．サービスの利用中にセッションの有効期間を超過すると，クライアントは最初から要求をやり直す必要がある．

Web アプリケーションの利用環境の拡大に伴い，クライアントの移動や通信環境の動的変化によって，サーバへのアクセスが中断する状況が増加している．そのため，サービス利用中にセッションの有効期間を超過する機会も増加し，クライアントはそのたびに最初から要求を繰り返さなければならない．その結果として Web アプリケーションの利便性が低下するという課題が生じている．この課題は，セッションの有効期間を無期限にすることで一応解決できる．しかし，セキュリティ向上やリソースの有効利用は必須であることを考慮すると，この解決策は現実的ではない．

本論文では，セッションの有効期間と状態を切り分けて管理する方式を提案する．提案方式により，セッションの有効期間が頻繁に超過するような環境下で，Web アプリケーションの利便性低下を防ぐことができることを示す．また，提案方式を実装した Web アプリケーションの性能を評価し，従来のセッション管理方式と変わらず適用可能な方式であることを示す．

2．利用環境の拡大で生じる従来方式の問題点

多くの Web アプリケーションでは，効率よく安全なアプリケーションを開発するため，J2EE や ASP など各ベンダーから提供されているセッション管理方式[1，2]を利用している．これら従来のセッション管理方式（図 1）では，クライアントごとに一意かつ任意なセッション ID を割り当て，クライアントがリクエストをするたびに，リクエストにセッション ID を付加して送信する．サーバはこのセッション ID によりクライアントを識別する．セッション ID のやり取りには，主に Cookie が使用されている[1, 2, 3]が，Cookie に非対応のクライアントの場合には URL や，hidden フィールドにセッション ID を埋め込む方法が用いられる．

サーバはクライアントの新しいリクエスト R に対して，セッション ID と 1 対 1 に対応するオブジェクト O をメモリ上に生成し，クライアントから送信されたデータをオブジェクト O に格納し，その状態を保持する．サーバは，オブジェクトのメモリ上の生存時間をセッションの有効期間として管理する．従って，従来方式では，セッションの有効期間と状態保持期間が同じになる．

セッションの有効期間を超過する機会が増加するモバイル・ユビキタス環境では，この特徴が Web アプリケーションの利便性低下の原因となっている．有効期間の超過によって，サーバ側で保持していた状態も同時に消失する．そのため，有効期間を超過した後にクライアントがリクエストを続行した場合，それまでのセッション中で実行した処理はすでに無効であり，送信したデータもサーバに存在しない．よって，クライアントは最初からリクエストをやり直して有効期間が超過する前の状態を復元しなければならず，Web アプリケーションの利便性は低下する．

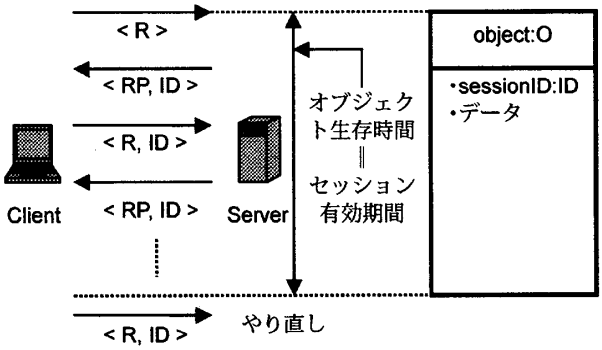

図 1 従来のセッション管理方式

3. 利便性低下を防ぐセッション管理方式

3.1 提案方式の概念

2 章で述べた課題を解決するため，セッションの有効期間とデータの保持状態をそれぞれ別のオブジェクトで管理する方式を提案する．提案方式を図 2 に示す．

クライアントの新しいリクエストに対して，サーバはメモリ上にセッションの有効期間を管理するオブジェクト P と，データを格納した状態保持オブジェクト S を生

†（株）日立東日本ソリューションズ

成する．1クライアントに対して2つのオブジェクトが対応するため，クライアントの識別には2つのIDをサーバ間とやり取りする．サーバ側では，状態保持オブジェクトのメモリ上での生存期間（状態保持期間）Tを，セッションの有効期間tよりも長く設定する．これによって，状態保持オブジェクトが生存している間は，有効期間超過のたびにクライアントとの間に新たなセッションを生成することができる．このとき，クライアントのセッション有効期間は見かけ上，状態保持期間に等しくなる．有効期間を超過した際のセッション切替えをサーバ側で行うことによって，クライアントは有効期間の超過により生じる要求の再送信を行わずに処理を継続できる．以上のことから，提案方式を採用することによって，状態保持期間中はWebアプリケーションの利便性低下を防ぐことができる．

従来方式を採用するWebアプリケーションでは，サーバのメモリ確保やセッションID盗聴によるなりすましのリスクを低減するために，セッションの有効期間を短く設定している．提案方式の場合も従来と同様，メモリ確保やセキュリティリスク低減のため，有効期間管理用のオブジェクトの生存期間を短く設定する．特にセキュリティについては，従来同様IDの盗聴を完全に防ぐことはできないが，クライアントの認証をセッション切替え時にも行うことでなりすましのリスクを低減できる．その上で，状態保持オブジェクトの生存時間をある程度長い期間に設定することによって，Webアプリケーションの利便性を維持することができる．

3.2 提案方式の実現に必要なクラス

提案方式を実現するために必要なクラスの構成を図3に示す．提案方式は，セッションの有効期間を管理するオブジェクトに，従来方式で使用するオブジェクトをそのまま流用し，それに状態保持のための3つのクラスを追加することで実現される．

Managerクラスは状態保持オブジェクトの生存期間を持ち，それをもとにクライアントごとに生成した状態保持オブジェクトの生存期間を管理する．Managerクラスは Webアプリケーションの初期化時にオブジェクトとして生成され，別スレッドで生存確認用のメソッドを定期的に実行する．

Collectionクラスは状態保持オブジェクトを生成・削除・取得するためのコンテナクラスである．Managerクラスと同様，Webアプリケーションの初期化時にオブジェクトとして生成される．

Stateクラスはクライアントの状態を保持するクラスである．Stateクラスは，クライアントごとの状態を識別するための SID と，削除可能かどうかという生存の有効性に関する情報を持つ．Stateクラスは新しいクライアントのリクエストごとにオブジェクトとして生成され，直近のリクエストからManagerクラスで設定された生存期間の間だけメモリ上に生存する．

3.3 提案方式によるセッション管理の流れ

提案方式を採用したときのセッション管理の流れを図4，5に示す．図4では，クライアントがサーバにログインした後の，通常の処理の流れを示す．図5では，セッション

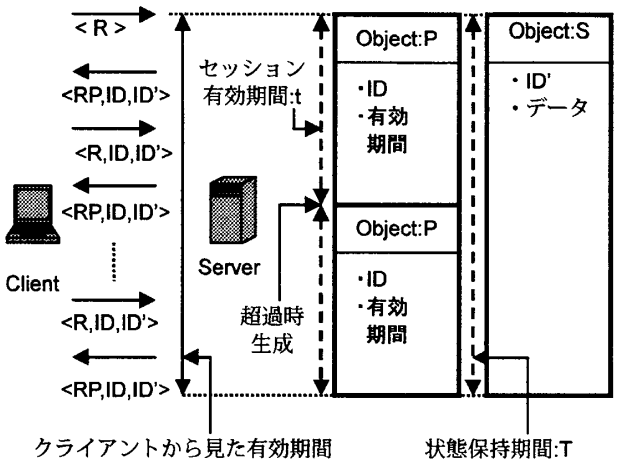

図2 提案するセッション管理方式

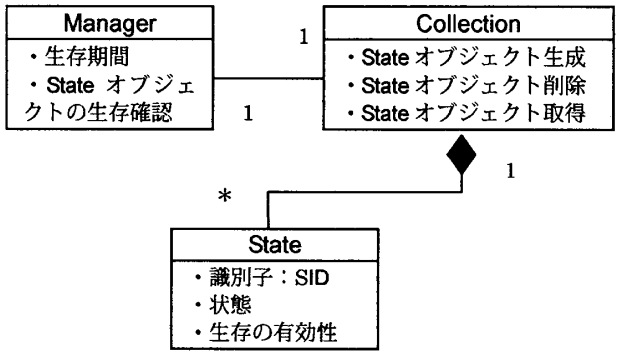

図3 提案方式のクラス構成

の有効期間を超過した後のリクエストで，新たにセッションを生成して処理を継続する際の処理の流れ示す．

クライアントがログインに成功した後，サーバのメモリ上に有効期間を管理するオブジェクトと，状態を保持するオブジェクトが生成される（図4の①）．サーバはこのクライアントからのリクエストを識別するために，各オブジェクトから発行された VID（有効期間管理オブジェクトの識別子）と SID（状態保持オブジェクトの識別子）をレスポンスとともにクライアントに送信する（図4の②）．クライアントはサーバへの以降のリクエストに対して，送信データとともに VID と SID を送信する（図4の③）．サーバでは，送信された VID をもとに有効期間管理オブジェクトの取得を試みることで，セッションが有効であるかを確認する（図4の④）．有効であれば，SID と合わせて状態保持オブジェクトの取得（図4の⑤）と送信データの保持（図4の⑥）を行う．以降のリクエストとレスポンスでは，生成したセッションが有効な間同様の処理が行われる．

図5は，既存のセッションが無効になった後の，クライアントのリクエストである．リクエストには，これまでサーバとやり取りしていた VID と SID が付加されている（図5の①）．サーバは VID をもとに有効期間管理オブジェクトの取得を試みる（図5の②）が，有効期間を超過しているために取得できない．しかしこの場合も，サ

ーバは SID と合わせて状態保持オブジェクトの取得を試みる（図5の③）．状態保持オブジェクトが生存しており，取得することができれば，新たなセッションの有効期間管理オブジェクトを生成する（図5の④）．次に，状態保持オブジェクトの持つ VID を新たに発行された VID1 で置換え，送信データを保持する（図5の⑤）．その後，サーバはクライアントに更新後の VID1 と SID を返す（図5の⑥）．以降の処理では，新しいセッションが有効な間，図4で示す手順と同様の手順でセッション管理を行う．

4. 性能評価

モバイル環境で利用する Web アプリケーションの構築で，提案方式が従来方式と比較して十分適用可能であることを示すため，提案方式，従来方式の両方式で Web アプリケーションを構築し，その性能を評価した．

3 章でも述べたように，提案方式は従来方式の拡張として実現できる．従来方式では，クライアントごとに1つのオブジェクトがメモリ上に生成されるが，提案手法ではそれに加えて 3.2 節で示したクラスのオブジェクトが生成される．従って，提案方式の採用により Web アプリケーションのメモリ使用量は，従来よりも増加することが予想される．本方式がユビキタス環境という限られたリソース上での Web アプリケーション構築にも適用されることを考慮すると，メモリ使用量が著しく多くなる場合には，従来方式に変わって本方式を適用することは困難になる．そのため，提案方式で Web アプリケーションのメモリ使用量がどの程度増加するのかを評価する必要がある．

また，提案方式ではサーバ側のセッション管理の処理が従来と比較して複雑化する．従って，従来と比較してレスポンスタイムが大きくなると予想され，その差が非常に大きい場合には，メモリと同様提案方式の適用は困難になる．そのため，本方式の採用によってレスポンスタイムにどの程度影響が出るのかを評価する必要がある．

メモリ使用量とレスポンスタイム以外の性能については，両方式で大きく異なると想定されるものはないため，以上の2項目について評価した．

評価用の Web アプリケーションは，ログイン後，商品を選択して買い物かごに入れ，数量を変更し，個人情報を送信して購入し，ログアウトするという仮想的なサービスである．利用環境が拡大する中で，Web アプリケーションが多様な OS 上で実行されていることから，実装には J2EE の技術である JSP および Servlet を用いた．提案方式で利用するセッションの有効期間管理オブジェクトには HttpSession オブジェクトを用いた．実験の測定環境を表1に示す．

Web アプリケーションのメモリ使用量については，以下の2つを比較する．

- 複数クライアントのアクセスによってオブジェクトが生成されたときの最大メモリ使用量 (従来方式)
- 複数クライアントのアクセスによって，セッション有効期間の管理オブジェクトと状態保持オブジェクトが生成されたときの最大メモリ使用量 (提案方式)

実験ではクライアント数が 100 と 500 の場合で比較した．実際の運用では，実行中にオブジェクトの生成・削除 が行われるが，構築に必要なメモリ量を評価するためには，全てのオブジェクトがメモリ上にあるときの最大使用量を測定すればよい．そのため，セッションの有効期間と状態保持期間は，最大メモリ使用量を測定可能なほど十分な時間を設定している．また，送信データ保持によるメモリ消費は両方式で同じであるため，各クライアントがデータを送信しないときの使用量を測定する．

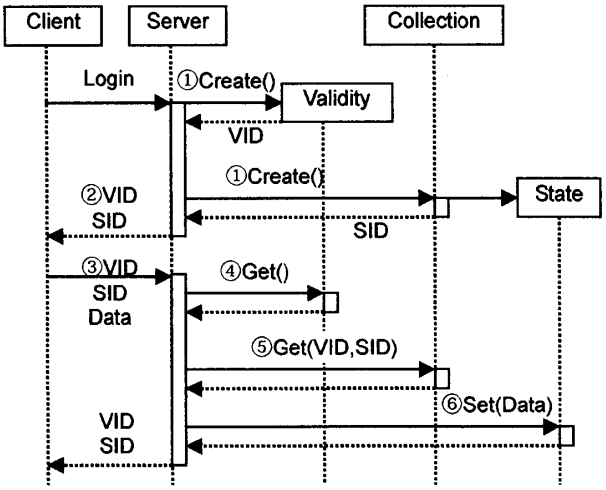

図4　提案方式のセッション管理の流れ (通常)

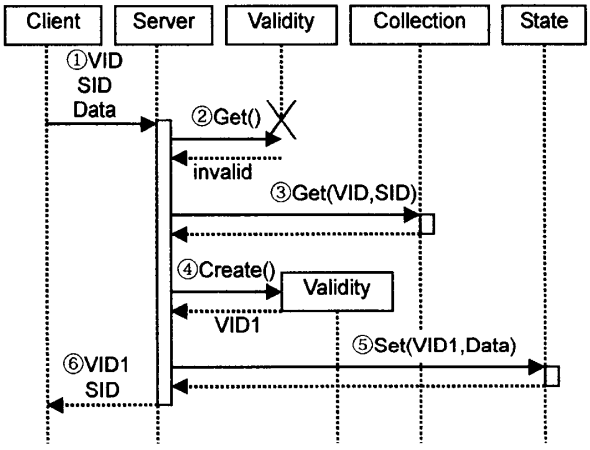

図5　提案方式のセッション管理の流れ (処理継続)

表1　実験の測定環境

Server	
CPU	Pentium4　2.6GHz
Memory	1GB RAM
OS	Windows XP　(SP1)
Web Server	Apache2.0.54
Web Container	Tomcat4.1.31
Java	JDK1.5.0

Client	
Browser	IE6.0
Access Tool	OpenSTA1.4.3

レスポンスタイムについては，複数クライアントがWebアプリケーションに同時アクセスしてログインし，商品を購入してログアウトするという一連の処理の平均レスポンタイムを測定する．各クライアントは一連の処理で7回のリクエストを行う．各リクエストのレスポンスタイムの平均値を更にクライアント数で平均して比較する．各リクエストでサーバから返されるデータ量は平均で約2kBである．両方式でクライアント数が10と20の場合を測定した．

まず，Webアプリケーションのメモリ使用量の測定結果を図7，8に示す．図7はクライアント数が100のとき，図8は500のときを示している．最大メモリ使用量は，100クライアントのとき提案方式で9,469kB，従来方式で9,046kB，その差は423kBであり，500クライアントのとき提案方式で11,100kB，従来方式で10,822kB，その差は278kBであった．100クライアントがアクセスする過程では，Webアプリケーションのメモリ使用量は提案方式のほうが，ほぼ常に少し多い傾向にある．500クライアントでは，ガベージコレクションの対象となるメモリ量やタイミングが異なってくるため，従来方式の方が，メモリ消費量が大きくなる場合がある．Webアプリケーション全体で使用するメモリ量が約6〜11MBであることから判断すると，どちらのクライアント数でも，提案方式の採用によって余分に使用されるメモリ量は十分小さいと言える．

次に，Webアプリケーションの平均レスポンスタイムの結果を表2に示す．同時アクセスするクライアント数によらず提案方式の方が高い値となっており，かつクライアント数が増加するほど平均レスポンスタイムも増加する傾向がある．しかし，両方式の平均レスポンスタイムの差は，10アクセスで6.6ミリ秒，20アクセスで21.0ミリ秒の差となっており，サーバの性能を大きく低下させるほどの差ではない．よって，提案方式によるレスポンスタイムへの影響も小さいと言える．

以上の結果から，提案方式を採用してWebアプリケーションを構築しても，従来方式と比較してWebアプリケーションの性能は大きく変わらないと言える．そのため，モバイル環境で利用される従来方式のWebアプリケーションを，提案方式で置き換えることによって，性能を大きく劣化させずに利便性の低下を防ぐことが可能となる．

5. おわりに

本論文では，モバイル環境という従来よりも拡大された環境でWebアプリケーションを利用する際に，Webアプリケーションの利便性を維持するための新たなセッション管理方式を提案した．提案方式では，セッションの有効期間とクライアントの状態をそれぞれ別のオブジェクトとしてサーバで管理する．これにより，セッションの有効期間を頻繁に超過してもクライアントは処理を継続でき，Webアプリケーションの利便性を維持することができる．

また，両方式で構築したWebアプリケーションサーバの性能を比較することにより，上記環境において提案方式が従来方式と変わらず適用可能であることを示した．

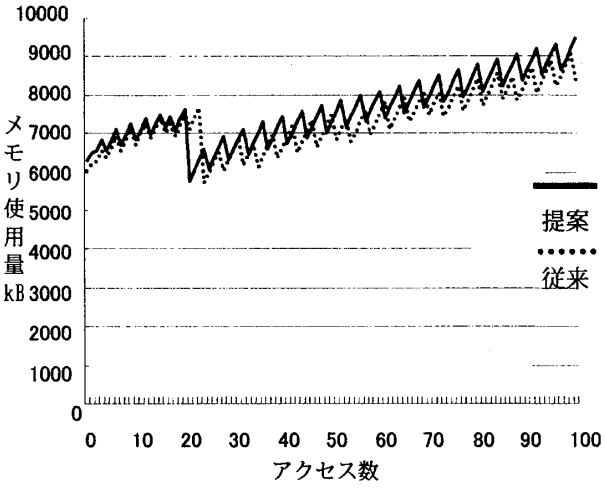

図7　メモリ使用量（100クライアント）

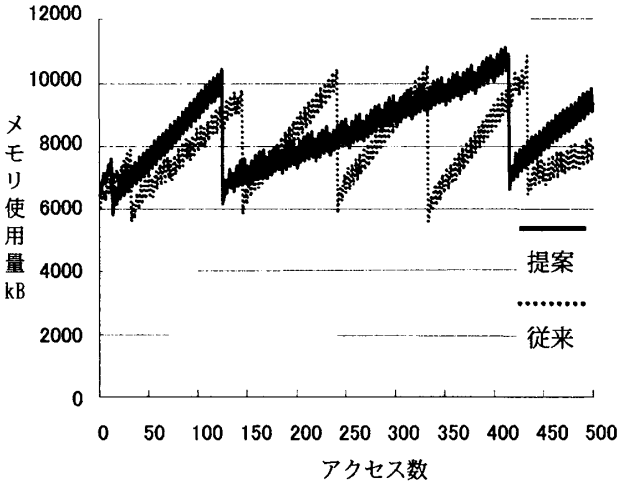

図8　メモリ使用量（500クライアント）

表2　平均レスポンスタイム（ミリ秒）

Client数	従来	提案	提案−従来
10	239.8	246.4	6.6
20	213.3	234.3	21.0

参考文献

[1] Danny Coward, Yutaka Yoshida , “Java Servlet Specification Version 2.4”, http://java.sun.com/products/servlet/download.html

[2] Michael P. Levy, “ASPとWebセッション管理”, http://www.microsoft.com/japan/msdn/web/server/asp/aspwsm.asp

[3] 栗原まり子，清原良三，橘高大造，木野茂徳，”携帯電話向けセッション管理方式の検討”，マルチメディア，分散，協調とモバイルシンポジウム，pp. 471-474, 2002.