モバイル環境で利用するWebアプリケーションのセッション管理手法

Session Management Method for Web Applications on Mobile Network

宗形　聡　　Munakata Satoshi

Web アプリケーションの利用環境は，有線 LAN に接続した端末からモバイル端末へと拡大している。Web アプリケーションのサービスは，サーバとの複数 Web ページにおける対話処理を通してクライアントに提供されるため，サーバはクライアントの識別や対話中の状態を保持するセッション管理機能を持つ。セッションの有効期間は，セキュリティなどを考慮して短く設定される。利用環境の拡大に伴い，クライアントが移動などによってサービスを利用できない間に有効期間を超過し，再開のたびに対話処理をやり直す状況が増加しており，その結果，Web アプリケーションの利便性が低下するという問題が生じている。そこで，従来サーバ側の１つのオブジェクトで管理されていたセッションの有効期間と状態を，それぞれ別に管理する手法を確立した。本手法により，従来と同等のセキュリティ水準を実現しながら Web アプリケーションの利便性を維持することが可能となる。

１．はじめに

従来 Web アプリケーションは，有線 LAN に接続したデスクトップ PC からの利用が中心であった。しかし，近年の通信利用動向に関する調査データ [1] によると，個人のインターネット利用端末はモバイル化が進展し，また，社内ネットワークを構築している企業の約半数が外出先など社外からのアクセス手段を確保している。さらに，無線 LAN を利用した社内ネットワークの構築も普及してきている。このような状況の下，ノート PC や移動端末から無線 LAN や移動体通信網を通して Web アプリケーションを利用する機会が増加している。

Web アプリケーションで提供されるサービスの多くは，クライアントと Web アプリケーションのサーバとの複数 Web ページにわたる対話処理によって実施される。クライアントとサーバ間の通信プロトコルは通常 HTTP であるが，HTTP はステートレスなプロトコルであるため，複数ページで対話処理を行うにはサーバ側でクライアントのセッションを管理する必要がある。

セッション管理では，クライアントの識別と対話中の状態保持を行う。セッションの有効期間には，セッション ID を盗聴されたときのセキュリティ脅威や，クライアントが明示的にセッションを終了しない場合のメモリ解放などを考慮して短い期間が設定される。

その結果，Web アプリケーションの利用環境の拡大に伴い，クライアントが移動などにより長時間サービスを利用できない間にセッションの有効期間を超過する（セッションタイムアウト）機会が増加している。セッションタイムアウト後にクライアントがページ要求を再開しても，最初から対話処理をやり直す必要があるため，セッションタイムアウトの増加は Web アプリケーションの利便性を低下させるという問題をもたらしている。この問題は，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が，この場合サーバのセキュリティ向上やリソースの有効利用はできない。これらの事項は Web アプリケーションにとって必須の要求事項であるため，より良い解決策が必要とされている。

セッション管理の分野では，セキュアな Web アプリケーションを構築するための，セッション ID 生成や送受信の方法などが考案されており，それらは J2EE や ASP などの技術で提供されている [2),3)]。近年では，リソース

が制約されたモバイル端末でのセッション ID の送受信や[4]，クロスサイト・スクリプティングなどのセキュリティ攻撃に対する脅威を軽減した Web アプリケーション構築に関する研究開発が行われている[5]。しかし，本報告で扱うような，タイムアウトに伴う利便性低下の問題についての研究は見当たらない。

セッションの有効期間を無期限にすることなく利便性の低下を防ぐには，タイムアウトした後のページ要求でクライアントが対話処理を継続できる必要がある。そのためには，対話処理の再開時にそれまでのセッションで対話した状態が必要となる。しかし，従来の手法ではセッションタイムアウトと同時に状態は失われる。そこで，セッションの有効期間を超過して状態を保持するために，状態をセッションの有効期間と切り分けて保持する方法が考えられる。この方法を実現する手段としては，サーバのメモリ上に状態保持用のデータ格納領域を新たに確保する手段や，クライアントがローカルに状態を保持する手段がある。

クライアントがローカルに状態を保持する場合には，リッチクライアントを実現する Applet を用いて Web アプリケーションを構築する。クライアントはサーバから Applet の class ファイルをダウンロードして実行し，ローカル端末のメモリ上に対話中の状態を保持する。しかしこの方法では，クライアント側の Java 実行環境や使用メモリ量などの制約を考慮する必要がある。また，クライアントが Applet を一旦終了してしまうと状態は失われるため，途中でブラウザを閉じることや，ログオフすることはできない。

一方，サーバ側で状態を保持する場合には，上記のようなクライアント端末の要件やブラウザ操作の制約はなく，クライアントがローカルに状態を保持する手段よりも有用な解決手段となる。

以上のことから，本報告では，状態保持用のデータ格納領域をセッションの有効期間と切り分けてサーバのメモリ上に確保するセッション管理手法を提案する。提案手法により，セキュリティ向上やリソースの有効利用を考慮しながら Web アプリケーションの利便性を維持できることを示す。また，提案手法を適用した場合のサーバ性能を従来手法と比較することにより，提案手法の有効性を示す。

2．従来のセッション管理手法の概略と課題

Web アプリケーションの開発には，ASP やサーバサイド Java などの技術が用いられることが多い。これらの技術では，セッションを管理するための機能があらかじめ提供されており，開発者はその提供された機能を利用して Web アプリケーションを利用するクライアントのセッション管理を効率的に実現できる。

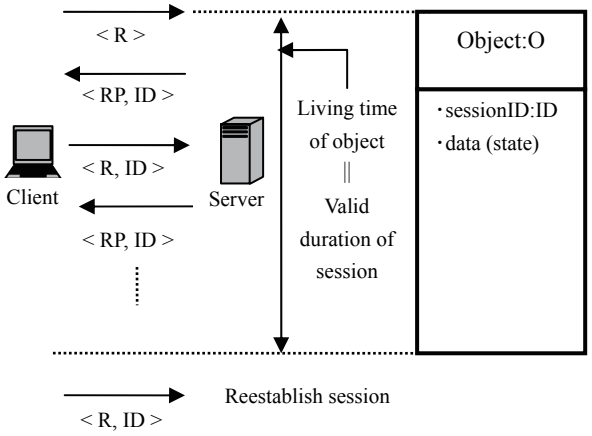

図 1　従来のセッション管理手法

これら既存のセッション管理機能では，ページ要求をするクライアントを識別するために，セッション ID を使用する。また，クライアントのセッション中の状態を保持するために，サーバのメモリ上にオブジェクトを生成する。セッション ID とオブジェクトは 1 対 1 に対応しており，それぞれクライアントの数だけ生成される。従来のセッション管理手法を図 1 に示す。

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（ID）とオブジェクト（O）を生成し，レスポンス（RP）に ID を付加して送信する。クライアントは次のページ要求時にサーバから送信されたセッション ID をリクエストに付加して送信する。サーバは，クライアントから送信された ID に対応するオブジェクト（O）を取得し，送信データをオブジェクトに格納して状態を保持する。以降，セッションが継続する間は同様の処理が繰り返され，複数ページにわたる対話処理が実施される。クライアントとサーバ間でのセッション ID の送受信には，主に Cookie が使用されている[2),3)]が，Cookie に非対応のクライアントの場合には，セッション ID を URL や hidden フィールドに埋め込む方法が用いられる。

従来手法では，サーバはセッションの有効期間をオブジェクト (O) のメモリ上での生存期間として管理する。このため，クライアントのセッション中の状態保持期間は，セッションの有効期間に等しくなる。したがって，

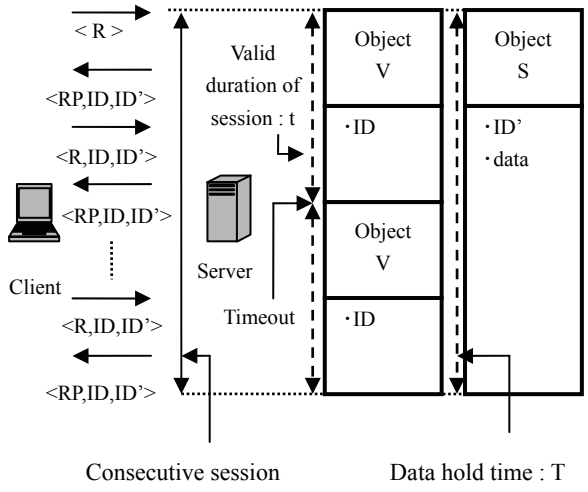

図 2　提案するセッション管理手法

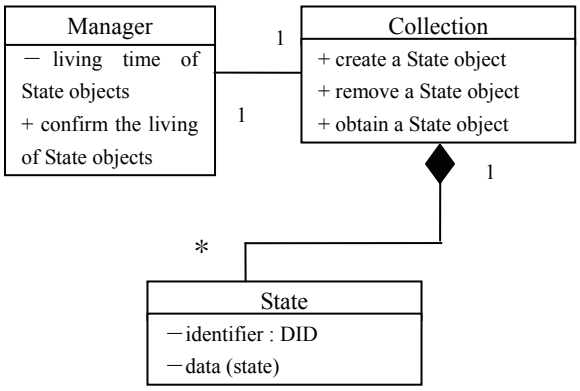

図 3　提案手法のクラス構成

クライアントがサーバのレスポンスを受信してからオブジェクト（O）の生存期間（すなわちセッションの有効期間）を超えてリクエストを中断した後にページ要求を再開しても，クライアントがセッション中にそれまで実行した処理や送信したデータは無効になっており，対話処理の継続はできない。この場合，クライアントは対話処理を最初からやり直してセッションの有効期間を超過する前の状態を復元する必要がある。

従来手法のように，セッション管理にセッション ID を用いる場合，セッション ID が第三者に盗聴されると，サーバはセッションハイジャックなどのセキュリティ脅威にさらされる。このため，サーバ側ではセッションの有効期間を短くしてこの脅威にさらされるリスクを低減する。また，クライアントが明示的にセッションを終了しない場合に，オブジェクトがメモリ上に生存し続けることを防ぐという観点からも，セッションの有効期間を短く設定する必要がある。

以上のことから，移動などによりページ要求を長い時間中断するモバイル環境では，短い有効期間のために中断中のセッションタイムアウトの機会が増加し，再開するたびに対話処理のやり直しが必要となる。その結果，Web アプリケーションの利便性は低下する。

3．モバイル環境向けセッション管理手法

3.1　提案手法の概略

Web アプリケーションの利便性低下を防ぐために，従来手法で 1 つのオブジェクトで管理されていたセッションの有効期間と送信データを，それぞれ別のオブジェクトで管理する手法について説明する。本報告で提案するセッション管理手法を図 2 に示す。

サーバはクライアントの新しいリクエスト（R）に対して，セッションの有効期間を管理するオブジェクト（V）と，送信データを格納する状態保持オブジェクト（S）をメモリ上に生成する。また，サーバはクライアントを識別するために 2 つの ID を生成し（ID，ID'），レスポンス（RP）に付加して送信する。ID はオブジェクト（V）と ID'はオブジェクト（S）とそれぞれ 1 対 1 に対応する。サーバは，クライアントの以降のリクエストに付加される 2 つの ID をもとに対応するオブジェクト（V），（S）を取得し，セッションが有効かどうかの検証や，送信データの保持を行う。

オブジェクト（V），（S）は従来と同様メモリ上での生存期間を持つ。オブジェクト（V）の生存期間はセッションの有効期間（t）であり，オブジェクト（S）の生存期間は状態保持期間（T）である。提案手法では，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とにより，有効期間の超過によってセッションがタイムアウトしても，オブジェクト（S）が生存している間は，サーバは新たにオブジェクト（V）を生成する。これによって，クライアントはセッションがタイムアウトした後にページ要求をしても，新しいセッションで対話処理を継続することができる。したがって，提案手法により状態保持期間中は Web アプリケーションの利便性低下を防ぐことができる。

提案手法は従来と同様に ID を用いるセッション管理手法であるが，セッションの有効期間を管理するオブジェクト（V）の生存期間を従来手法と同程度に短く設定し，タイムアウト後のページ要求で新しくオブジェクト（V）が生成される際にクライアントの認証を実施することによって，ID 盗聴時のセキュリティリスクを従来と

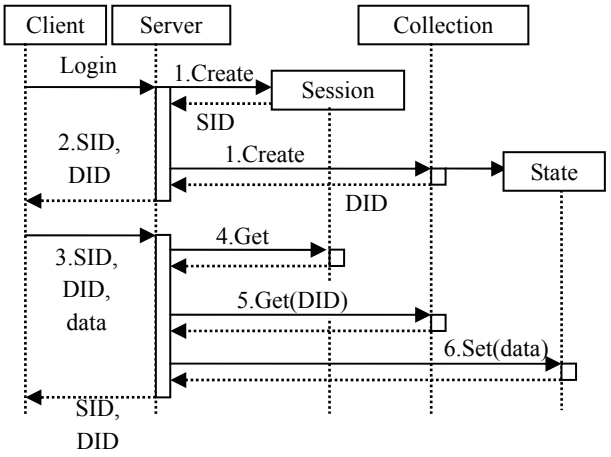

図 4　通常のセッション管理の流れ

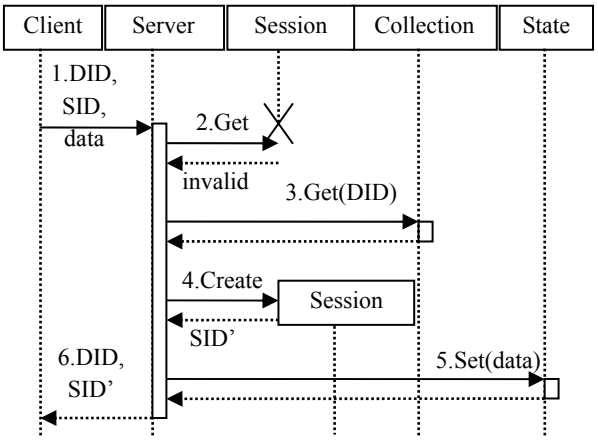

図 5　タイムアウト時のセッション管理の流れ

同様の水準まで低減できる。さらに，利便性の維持に適切な時間を状態保持期間に設定することによって，従来と同等のセキュリティ水準を保ちながら，Web アプリケーションの利便性を維持することができる。実際に設定される状態保持期間やセッションの有効期間の長さは，構築する Web アプリケーションの仕様や提供するサービスの特徴などを考慮して決定される。

また，セッションの有効期間と状態保持期間に有限な値を設定することにより，クライアントがセッションを明示的に終了しなくてもオブジェクトのメモリは解放される。

3.2　提案手法の実現方法

まず，提案手法を実現するために必要なクラス構成を図 3 に示す。セッションの有効期間を管理するオブジェクトに従来手法で使用したオブジェクトを流用し，図 3 に示す 3 つのクラスで状態保持機能を実現することによって，提案手法は実現できる。このように，提案手法は従来手法の拡張となっている。

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識別に用いる識別子の 1 つである DID を持つ。State クラスのオブジェクトは，前節のオブジェクト（S）のことであり，新しいクライアントのリクエストのたびに生成される。このオブジェクトは，対応するクライアントの最後のリクエストから Manager クラスで設定された生存期間の間だけメモリ上に存在する。

Manager クラスは状態保持オブジェクトのメモリ上での生存期間を持ち，この値をもとにクライアントごとに生成された状態保持オブジェクトの生存を管理する。Manager クラスのオブジェクトは Web アプリケーションの初期化時に生成され，Web アプリケーションの実行スレッドとは別のスレッドで状態保持オブジェクトの生存を管理するメソッドを定期的に実行する。

Collection クラスはすべての状態保持オブジェクトを持つコンテナクラスであり，サーバはこのクラスを通して状態保持オブジェクトを生成・取得・削除する。Collection クラスのオブジェクトは，Manager クラスと同様 Web アプリケーションの初期化時に生成される。

次に，提案手法によるセッション管理の流れを図 4 と図 5 に示す。図 4 にはクライアントが Web アプリケーションを利用するためにサーバにログインした後の，通常の対話処理の流れを示し，図 5 にはセッションの有効期間を超過した後のページ要求で，新しいセッションを生成して対話処理を継続するときの流れを示す。

通常の対話処理では，クライアントがログインに成功した後，サーバはオブジェクト（V）と（S）をメモリ上に生成する（図 4 の 1)。図 4 では，オブジェクト（V）を Session, オブジェクト(S) を State と表記している。このときサーバは，クライアントからの以降のリクエストを識別するために Session オブジェクトの識別子 SID と，State オブジェクトの識別子 DID を生成し，レスポンスに付加してクライアントに送信する（図 4 の 2)。クライアントは次ページのリクエストとともに送信データと 2 つの ID をサーバに送信する(図 4 の 3)。サーバは，送信された SID に対応する Session オブジェクトを取得し，セッションが有効かどうかを確認する（図 4 の 4)。セッションが有効であれば，サーバは DID をもとに State オブジェクトを取得し（図 4 の 5)，送信データを格納する（図 4 の 6)。

セッションの有効期間を超過した後にクライアントが

表 1　実験の測定環境

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

Client Side	
CPU	Pentium M 1.2GHz
Memory	752MB RAM
OS	Windows XP SP2
Browser	IE6.0

リクエストを再開した場合，サーバはこれまでのセッションでクライアントと送受信していた 2 つの ID を受信する（図 5 の 1)。サーバは SID をもとに Session オブジェクトの取得を試みるが，生存時間を越えた Session オブジェクトを取得することはできない（図 5 の 2)。このときサーバは，DID をもとに対応する State オブジェクトの取得を行う（図 5 の 3)。State オブジェクトを取得できれば，サーバは新たな Session オブジェクトを生成し（図 5 の 4)，送信データを State オブジェクトに保持する（図 5 の 5)。また，サーバはこのとき生成された Session オブジェクトの SID'と DID をレスポンスに付加してクライアントに送信する（図 5 の 6)。以降のクライアントのリクエストの識別には SID'と DID が使用され，新しいセッションが有効な間は図 4 の 3 以降に示す手順でセッションが管理される。

4．サーバの性能評価

提案手法の有効性を評価するため，従来手法と提案手法をそれぞれに適用して構築した Web アプリケーションのサーバ性能を比較した。比較する項目は Web アプリケーションのメモリ使用量と平均応答時間の 2 つである。

提案手法では，従来手法よりも状態保持のためのオブジェクトが余分に生成される。メモリ使用量は，このときのメモリリソースへの影響を評価するために測定する。また，提案手法では，セッション管理の処理が従来よりも複雑になる。平均応答時間は，このときのレスポンス性能への影響を評価するために測定する。上記 2 項目以外のサーバ性能については，提案と従来の両手法で同程度であると想定される。

構築した Web アプリケーションは仮想のオンライン

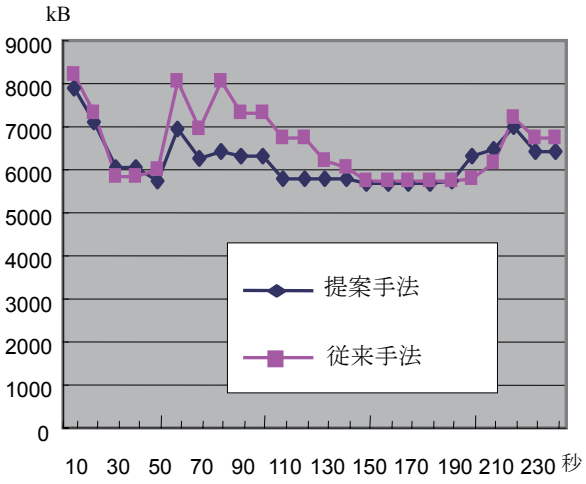

図 6　メモリ使用量の推移

表 2　平均応答時間（ミリ秒）

Client 数	従来	提案	提案－従来
10	152.1	169.5	17.4
20	191.3	238.1	46.8
40	300.1	304.4	4.3

商品購入サービスである。クライアントはログインした後，商品を選択してカートに入れ，数量の変更などをし，決済してログアウトする。Web アプリケーションは，その利用環境が拡大する中で多様な OS 上で動作することを考慮して，Servlet および JSP を用いて構築した。提案手法でセッションの有効期間の管理に使用する従来手法のオブジェクトは，HttpSession オブジェクトである。

Web アプリケーションのサーバとクライアントは，他ネットワークのトラフィックの影響を避けるため，閉じたローカルネットワークに接続して実験を行った。サーバとクライアントの詳細な測定環境を表 1 に示す。

メモリ使用量については，30 クライアントが同時にログインしてから複数ページで対話処理を行ってログアウトするまでの使用量を，10 秒ごとに測定する。運用に近い状況での使用量を測定するため，提案手法では，セッションの有効期間を 60 秒と設定し，各クライアントが途中で対話処理を中断してセッションがタイムアウトした後に処理を再開するように設定した。実験では，各クライアントはログイン後 80 秒から 200 秒までの 120 秒間リクエストを中断する。よって，ログイン後 140（80＋60）秒以降にセッションはタイムアウトする。従来手法でも同様にリクエストを中断するが，従来手法ではこの間セッションが有効であるように長い時間を有効期間

として設定した。これは，従来手法で有効期間を長くして提案手法と同程度の Web アプリケーションの利便性を実現した場合でメモリ性能を比較することにより，有効期間の長い従来手法よりもセキュリティ水準を高くできる提案手法の適用可能性を評価するためである。

両手法において各クライアントは，通常の対話処理で商品名や数量，決済時の個人情報などを入力した場合に想定される 300 文字程度の情報をサーバに送信する。メモリ使用量の測定は 10 回実施し，測定結果は 10 回の測定における平均値とした。

平均応答時間については，複数クライアントが同時に Web アプリケーションにアクセスし，ログインからログアウトまでに送信したリクエストに対する応答時間を測定する。各クライアントはログインからログアウトまでの 7 回リクエストを送信する。このとき，両手法において各クライアントの対話処理中にセッションタイムアウトはしない。平均応答時間は，7 回のリクエストに対する応答時間の平均を，さらにクライアント数で平均した値とした。

メモリ使用量の測定結果を図 6 に，平均応答時間の測定結果を表 2 に示す。図 6 より，対話処理を中断するログイン後 80 秒前後の時間帯（60～120 秒）では，提案手法よりも従来手法の方がメモリを使用している。1 クライアントがログインして生成される状態保持オブジェクトのメモリ使用量は約 5kB であり，30 クライアントでは 150kB となる。よって，対話処理中は提案手法のほうが 150kB 程度メモリを多く使用すると考えられる。しかしながら，このように従来手法の方がメモリを使用するという結果には，メモリ使用量が 10 回の測定の平均値であるため，各回の測定値のばらつきによる影響もあるが，JVM のメモリ管理状態にも要因があると考えられ，詳細については今後の課題である。

ログインして対話処理を開始した時間帯（0～50 秒）や，中断して処理を再開するまでの時間帯（130～200 秒），およびそれ以降の時間帯（210～240 秒）では，両手法でほぼ同程度か，提案手法が従来手法よりもメモリを使用している。その増分は，提案手法で対話処理を再開するログイン後 200 秒で最も大きく，500kB 程度である。この増分量も想定の 150kB より大きいが，対話処理再開時は新たに HttpSession オブジェクトを生成するため，その影響が出ていると考えられる。

一方，ログインからログアウトまで行われる対話処理全体の流れの中では，提案手法が従来よりも著しくメモリを使用することはないことが図 6 よりわかる。したがって，先に述べたメモリ管理に関する課題の原因を追究する必要があるものの，提案手法の適用によって従来と比較してサーバのメモリ性能が大きく劣化することはないと言える。

また，表 2 より複数クライアントが同時にリクエストすると，提案手法の方が従来よりもサーバのレスポンスタイムは増加することがわかる。しかし，その増分は高々数十ミリ秒であり，サーバの応答性能への影響も小さいと言える。

以上の実験結果から，クライアントの要件や操作に依存せず，モバイル環境から利用する Web アプリケーションの場合には，提案手法は従来手法に代わって十分適用可能であることが分かった。

５．おわりに

本報告では，従来のセッション管理手法を適用した Web アプリケーションをモバイル環境で利用する際に生じる，利便性の低下という問題を解決する新たなセッション管理手法を提案した。提案手法では，従来サーバ側で 1 つのオブジェクトで管理されていたセッションの有効期間とセッション中の状態を，それぞれ別のオブジェクトで管理する。サーバ側でセッションの有効期間よりも長い期間状態を保持することによって，有効期間を超過した後にクライアントがリクエストを再開しても対話処理が継続できる。従来と同様に有効期間を短く設定し，タイムアウト後のリクエスト時にサーバがクライアントを認証することによって，ID の盗聴に関して従来と同程度のセキュリティ水準を確保しながら，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，従来と提案の両手法を適用した Web アプリケーションのサーバ性能を比較することにより，クライアントの要件や操作に依存せず，モバイル環境からアクセスされる Web アプリケーションには，従来手法を提案手法に変更して適用可能であることを示した。

参考文献

1）総務省：平成 17 年通信利用動向調査，
http://www.johotsusintokei.soumu.go.jp/statistics/index.html

2）Danny Coward 他，“Java Servlet Specification Version2.4”，

http://java.sun.com.products/servlet/download.html
3）Michael P. Levy，”ASP と Web セッション管理”，http://www.microsoft.com/japan/msdn/web/server/asp/aspwsm.asp.
4）栗原まり子他，“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
5） Paul Johnston，“Authentication and Session Management on the Web“，http://www.giac.org/certified_professionals/practicals/GSEC/4206.php

宗形　聡　　2003 年入社
研究開発グループ
Web アプリケーションのセッション管理方式に関する研究開発に従事
munakata@hitachi-to.co.jp