ソフトウェア工学
ウェブアプリケーション

2022/12/19

海谷 治彦

目次

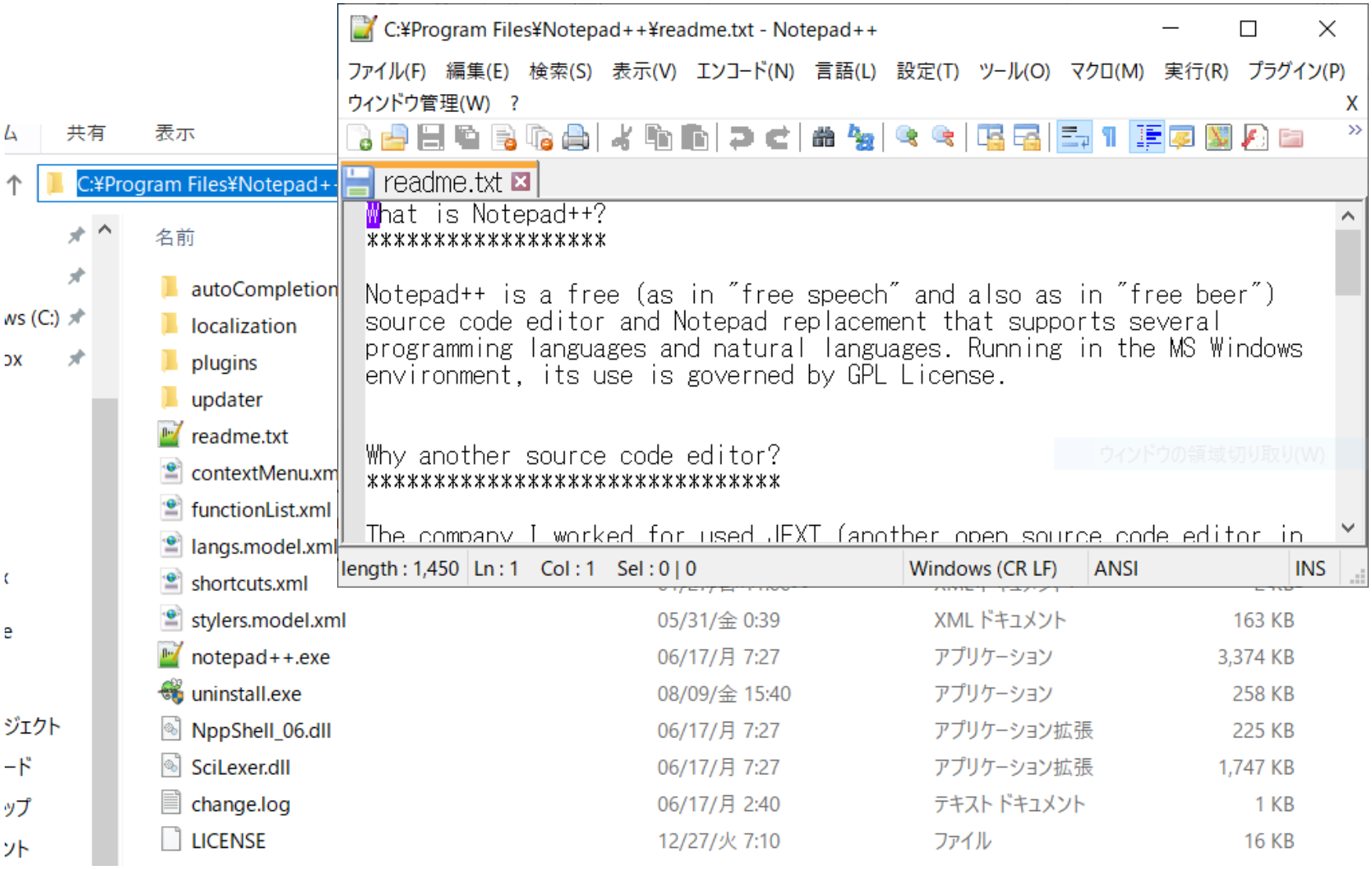
- まえおき・概要
- ウエブアプリケーションとは？
  - 普通のアプリやクライアントサーバーとの違い
- HTTP
- ウエブアプリケーションを支える言語等
  - HTML5, CSS3, JavaScript
  - Servlet/JSP, PHP, Ruby on Rails
- データの保持
- ウエブアプリケーションのシステム構成
  - ハードウェア
  - ソフトウェア
- セッション
- 簡単な歴史
- まとめ

普通のアプリケーションとは？

- 仕事，生活，遊び等に適用される(apply)情報システムをアプリケーション(application)と呼ぶ.

- 汎用OS上のプログラムとしてインストールされ，
- 実行することでプロセスとして動作する.
- 無料，有料に関係なく，実行するためには，事前にOS上にプログラム(の一部)をインストールしておく必要がある.

普通のアプリの例


C:¥Program Files¥Notepad++¥readme.txt - Notepad++
ファイル(F) 編集(E) 検索(S) 表示(V) エンコード(N) 言語(L) 設定(T) ツール(O) マクロ(M) 実行(R) プラグイン(P)
ウィンドウ管理(W) ?
readme.txt
What is Notepad++?
Notepad++ is a free (as in "free speech" and also as in "free beer") source code editor and Notepad replacement that supports several programming languages and natural languages. Running in the MS Windows environment, its use is governed by GPL License.
Why another source code editor?
length : 1,450 Ln : 1 Col : 1 Sel : 0 | 0
Windows (CR LF) ANSI INS
共有 表示
C:¥Program Files¥Notepad++
名前
autoCompletion
localization
plugins
updater
readme.txt
contextMenu.xml
functionList.xml
langs.model.xml
shortcuts.xml
stylers.model.xml 05/31/金 0:39 XML ドキュメント 163 KB
notepad++.exe 06/17/月 7:27 アプリケーション 3,374 KB
uninstall.exe 08/09/金 15:40 アプリケーション 258 KB
NppShell_06.dll 06/17/月 7:27 アプリケーション拡張 225 KB
SciLexer.dll 06/17/月 7:27 アプリケーション拡張 1,747 KB
change.log 06/17/月 2:40 テキスト ドキュメント 1 KB
LICENSE 12/27/火 7:10 ファイル 16 KB

ウエブアプリケーションとは？

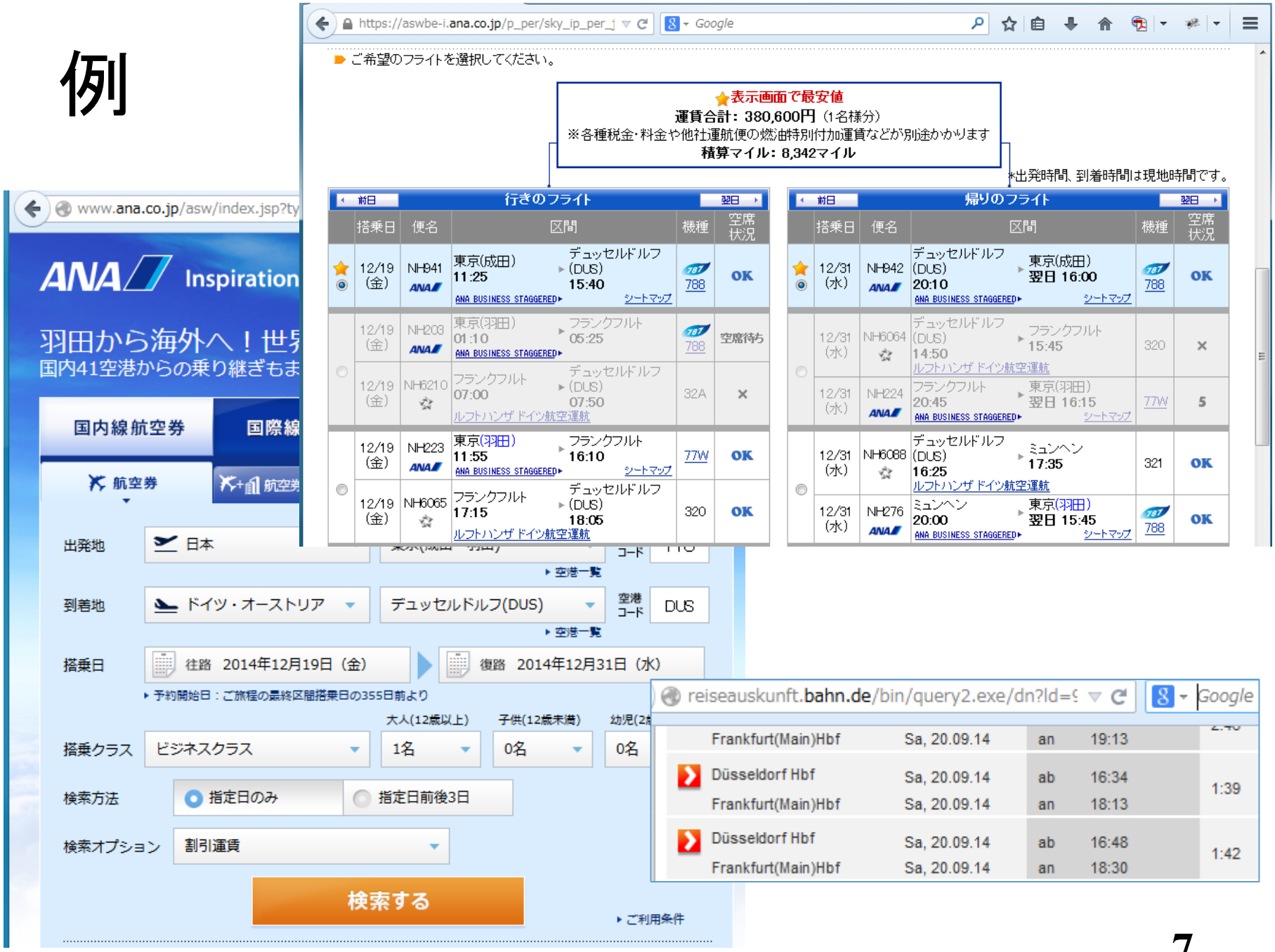
- ウエブブラウザ(EdgeやFirefox等)上で動作するアプリケーション.
- 事前にアプリケーションをインストールする必要は無い.
- 通常はブラウザ，サーバー側の多様なプロセスと連携して実行される.

具体的な例

- アマゾン等の買い物サイト
- 航空券，ホテル宿泊等の予約･購入サイト
- ブラウザゲー (今，何がメジャーなんだ？)
- 納税のサイト
- 不用品リサイクルの情報交換サイト

- 大学等の履修登録，受講支援，成績照会サイト

例


https://aswbe-i.ana.co.jp/p_per/sky_ip_per_j
Google
ご希望のフライトを選択してください。
表示画面で最安値
運賃合計：380,600円（1名様分）
※各種税金・料金や他社運航便の燃油特別付加運賃などが別途かかります
積算マイル：8,342マイル
*出発時間、到着時間は現地時間です。
行きのフライト
帰りのフライト
前日
翌日
搭乗日
便名
区間
機種
空席状況
12/19（金）
NH941
東京(成田) 11:25
デュッセルドルフ (DUS) 15:40
ANA BUSINESS STAGGERED
シートマップ
788
OK
12/19（金）
NH203
東京(羽田) 01:10
フランクフルト 05:25
ANA BUSINESS STAGGERED
788
空席待ち
12/19（金）
NH6210
フランクフルト 07:00
デュッセルドルフ (DUS) 07:50
ルフトハンザ ドイツ航空運航
32A
×
12/19（金）
NH223
東京(羽田) 11:55
フランクフルト 16:10
ANA BUSINESS STAGGERED
シートマップ
77W
OK
12/19（金）
NH6065
フランクフルト 17:15
デュッセルドルフ (DUS) 18:05
ルフトハンザ ドイツ航空運航
320
OK
12/31（水）
NH942
デュッセルドルフ (DUS) 20:10
東京(成田) 翌日 16:00
ANA BUSINESS STAGGERED
シートマップ
788
OK
12/31（水）
NH6064
デュッセルドルフ (DUS) 14:50
フランクフルト 15:45
ルフトハンザ ドイツ航空運航
320
×
12/31（水）
NH224
フランクフルト 20:45
東京(羽田) 翌日 16:15
ANA BUSINESS STAGGERED
シートマップ
77W
5
12/31（水）
NH6088
デュッセルドルフ (DUS) 16:25
ミュンヘン 17:35
ルフトハンザ ドイツ航空運航
321
OK
12/31（水）
NH276
ミュンヘン 20:00
東京(羽田) 翌日 15:45
ANA BUSINESS STAGGERED
シートマップ
788
OK
www.ana.co.jp/asw/index.jsp?ty
ANA Inspiration
羽田から海外へ！世界
国内41空港からの乗り継ぎもま
国内線航空券
国際線
航空券
出発地
日本
到着地
ドイツ・オーストリア
デュッセルドルフ(DUS)
空港コード
DUS
空港一覧
搭乗日
往路 2014年12月19日（金）
復路 2014年12月31日（水）
予約開始日：ご旅程の最終区間搭乗日の355日前より
大人(12歳以上)
子供(12歳未満)
幼児(2
搭乗クラス
ビジネスクラス
1名
0名
0名
検索方法
指定日のみ
指定日前後3日
検索オプション
割引運賃
検索する
ご利用条件
reiseauskunft.bahn.de/bin/query2.exe/dn?ld=
Google
Frankfurt(Main)Hbf Sa, 20.09.14 an 19:13
Düsseldorf Hbf Sa, 20.09.14 ab 16:34
Frankfurt(Main)Hbf Sa, 20.09.14 an 18:13
1:39
Düsseldorf Hbf Sa, 20.09.14 ab 16:48
Frankfurt(Main)Hbf Sa, 20.09.14 an 18:30
1:42

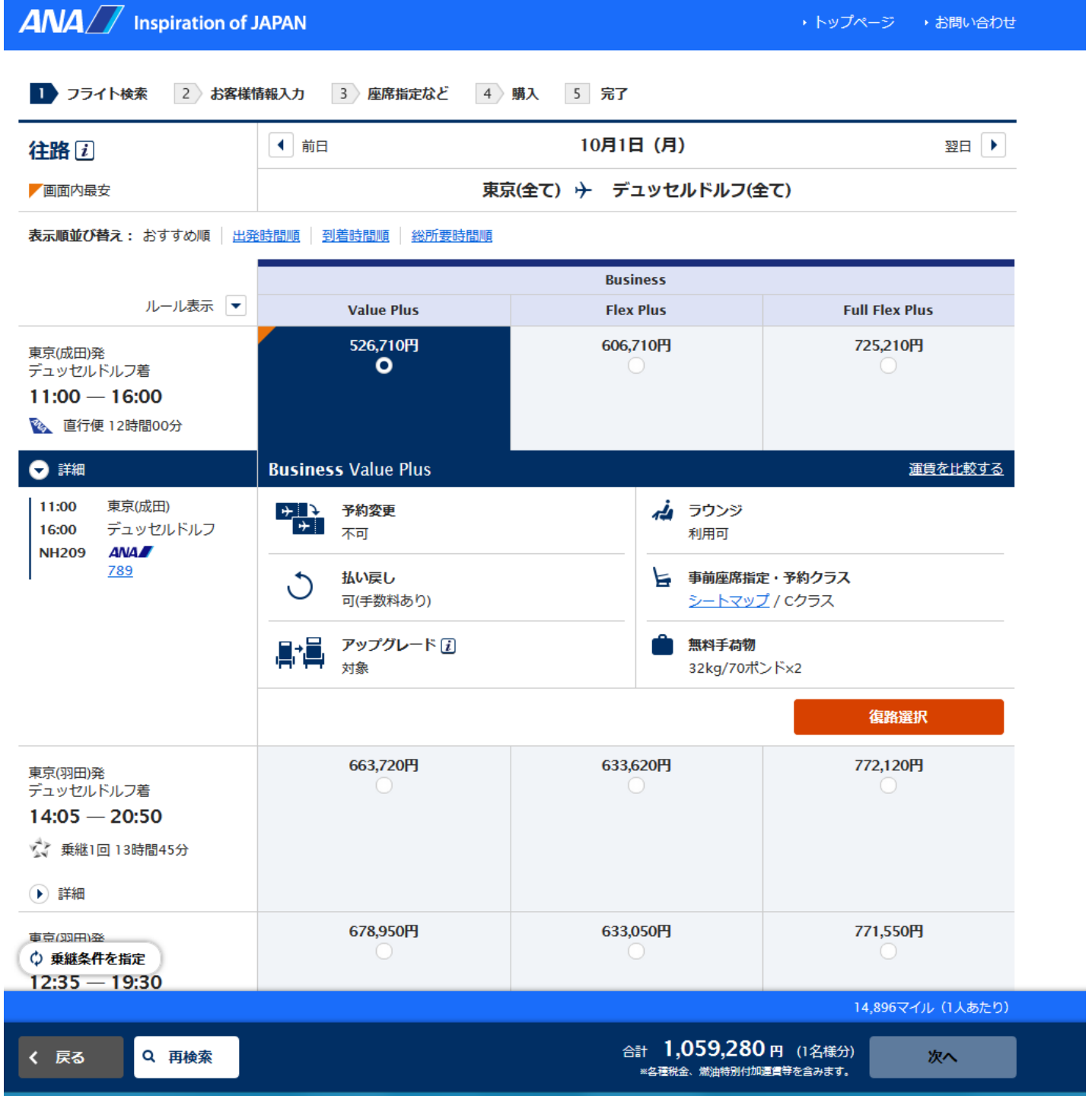
クライアント・サーバーとの違い

- ウエブアプリケーションはクライアント・サーバー コンピューティングの一種である.
- 以下の二点を限定したものと考えられる.
  - クライアント: ウエブ・ブラウザ
  - 主な通信プロトコル: HTTP HTTPS

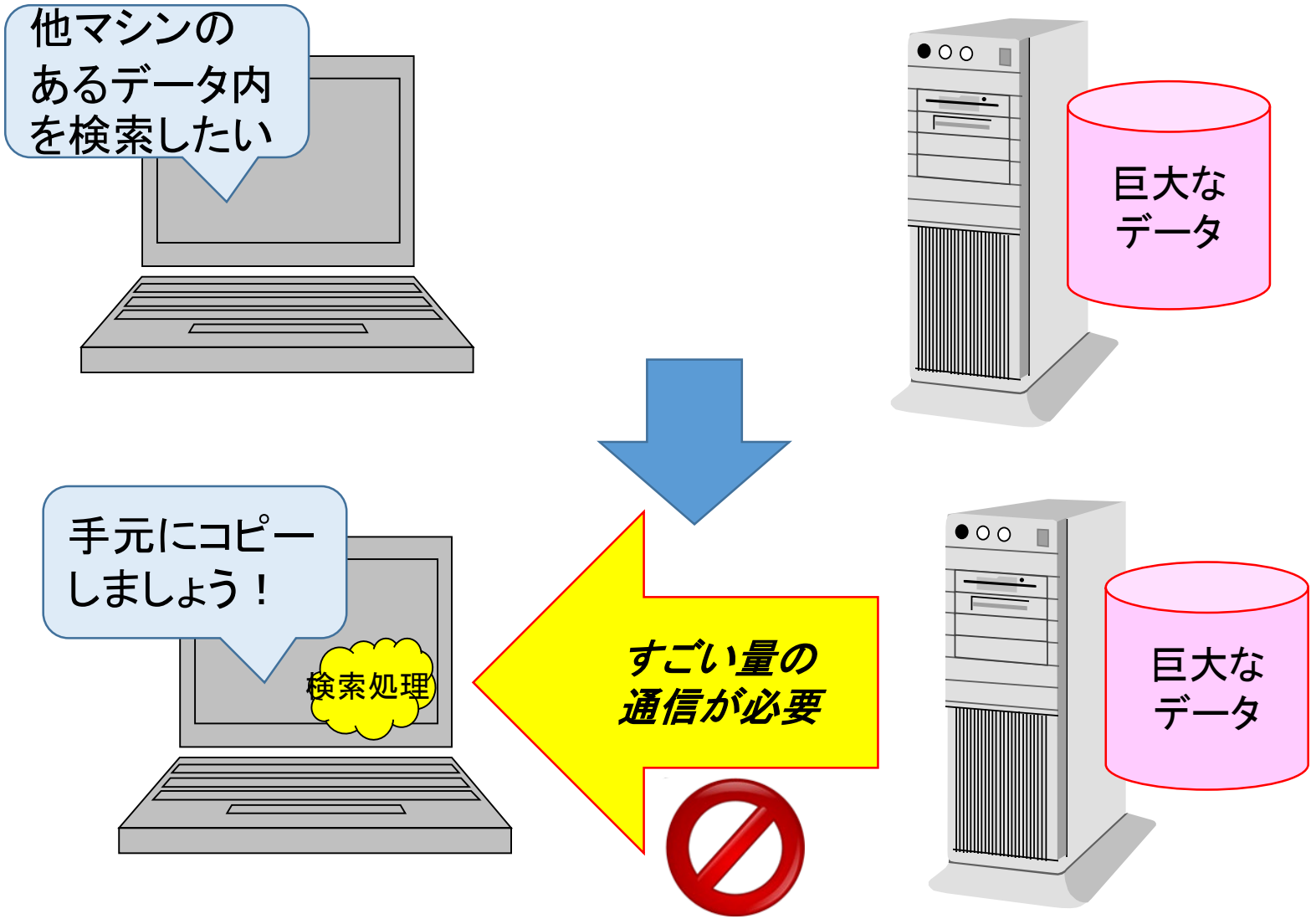
- 一般的なクライアント・サーバーでは，上記の二点の限定は無い.
  - クライアント: 独自ソフトでもよい.
  - 通信プロトコル: TCP上に独自のプロトコルを使う場合が多いが，TCP/IPである必要すらない.

ウェブアプリケーションの特徴

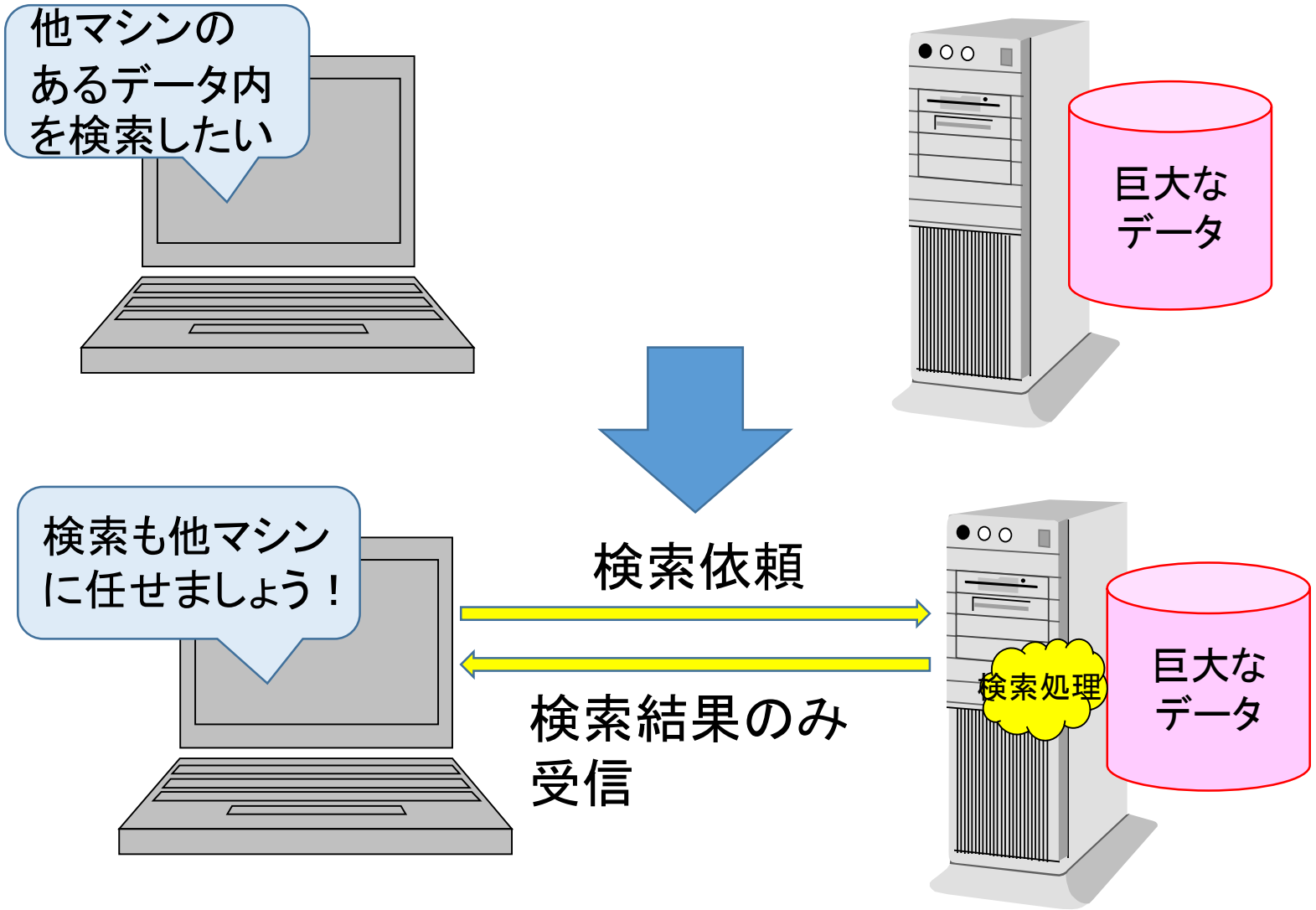
- インストール等の前準備が利用者にとっては不要
  - 一般的なクライアントサーバーの場合，クライアントの準備が必要
- アプリケーションの機能更新が比較的楽
  - 同じウエブアプリ(サイト)でも，再訪問するとがらっとかわっていることがある.
- 利用者側の端末が比較的非力でも利用可能
  - 基本，クライアントはユーザー入力取得，サーバーへの送受信，結果の表示をしているだけ.
- 利用者側で大規模なデータをかかえなくて済む


ANA Inspiration of JAPAN
トップページ
お問い合わせ
1 フライト検索
2 お客様情報入力
3 座席指定など
4 購入
5 完了
往路
画面内最安
前日
10月1日（月）
翌日
東京(全て) ✈ デュッセルドルフ(全て)
表示順並び替え： おすすめ順 | 出発時間順 | 到着時間順 | 総所要時間順
ルール表示
Business
Value Plus
Flex Plus
Full Flex Plus
東京(成田)発
デュッセルドルフ着
11:00 — 16:00
直行便 12時間00分
526,710円
606,710円
725,210円
詳細
Business Value Plus
運賃を比較する
11:00 東京(成田)
16:00 デュッセルドルフ
NH209 ANA
789
予約変更
不可
ラウンジ
利用可
払い戻し
可(手数料あり)
事前座席指定・予約クラス
シートマップ / Cクラス
アップグレード
対象
無料手荷物
32kg/70ポンド×2
復路選択
東京(羽田)発
デュッセルドルフ着
14:05 — 20:50
乗継1回 13時間45分
詳細
663,720円
633,620円
772,120円
乗継条件を指定
12:35 — 19:30
678,950円
633,050円
771,550円
14,896マイル（1人あたり）
戻る
再検索
合計 1,059,280 円 (1名様分)
※各種税金、燃油特別付加運賃等を含みます。
次へ

ウエブアプリが便利な例 1/2



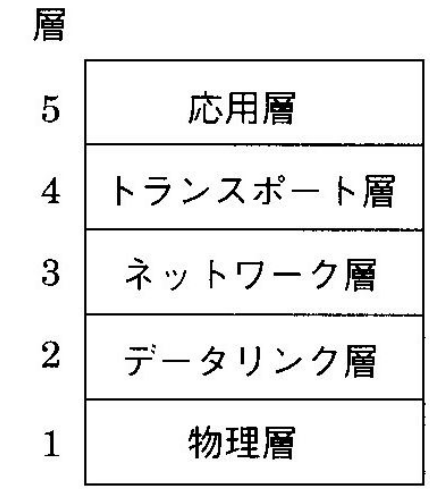
ウエブアプリが便利な例 2/2



プロトコル Protocol

- 「通信規約」のこと.
- 通信する異なる者(物)同士が，予め送受信するデータの意味の合意をとっておくこと.
  - 人間でいえば，日本語，英語等の話し言葉がプロトコルといえる.
- プロトコルは電気信号等の低レベルのものと，人間同士の会話のように意味のある高レベルのものとがある.

復習: 五階層モデル



OSIの7階層モデルの第5層と
第6層を除いたもの

図 11.1　ネットワークアーキテクチャの５階層モデル

- 低いプロトコルから高いプロトコルまで重ねたもの.
- 実体としては，物理層の電気的な信号しか無いが，
- それらを組み合わせることで，より意味のあるプトロコルを構成できる.

復習: 人間の会話との大雑把な対比

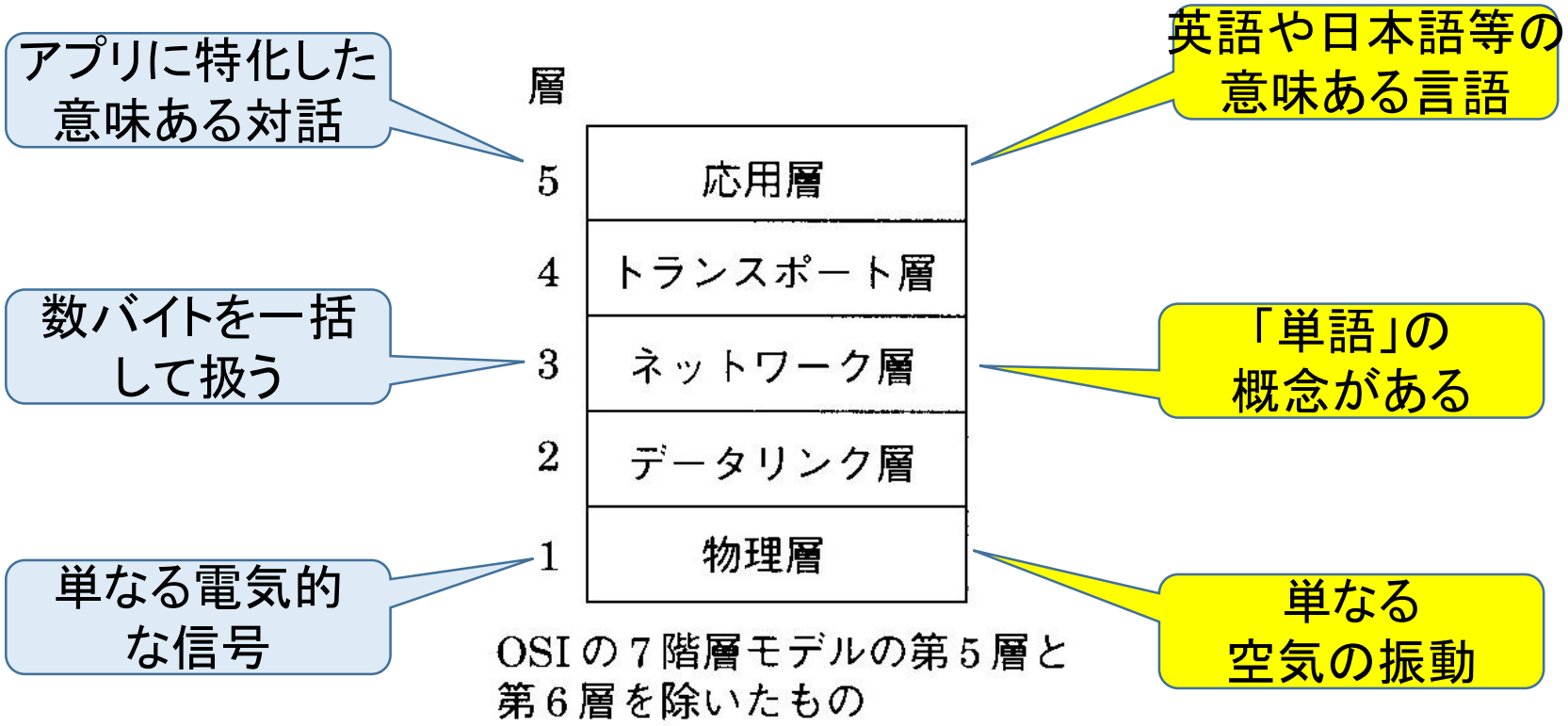
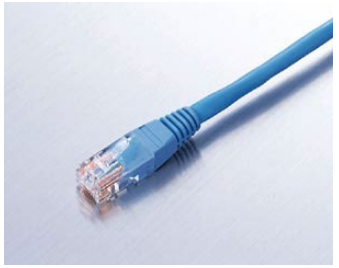


図 11.1　ネットワークアーキテクチャの５階層モデル

主な具体的なプロトコル



- MAC (データリンク層)
  - イーサネット等で採用されている方式
- IP (ネットワーク層)
  - 20~60バイトのデータを一組(パケット)として送る.
  - パケットが事故で消える場合があっても再送等はしない.
- TCP (トランスポート層)
  - パケットの再送機構を含むため，データが必ず送り先に順番に届くことを保証できる.
- UDP (トランスポート層)
  - データの消失，重複，順番逆転がありうるが，多数の相手への同時送信(ブロードキャスト)ができる.
- HTTP (アプリケーション層)
  - ウエブブラウザ等が表示データの送受信に使うプトロコル.

HTTPの詳細

- Hyper Text Transfer Protocol
- 基本的に以下のやりとりの対となる
  - クライアント(ブラウザ)側から「リクエスト」を送る
  - サーバー側から「レスポンス」が返る
- レスポンス，リクエストともに内容はテキストファイルである.
  - よって，直接，人間が目視で読むことが可能.

- HTTPレベルでは，リクエストに対するレスポンスが対応付いているだけである.
- 同じブラウザからの連続したリクエストである等の一連の処理(セッション)の識別はプログラムによって，プログラマが管理しないといけない.

もっとも基本的な例

- ブラウザが見たいページをリクエストして，
- そのページのデータがレスポンスとして返ってくる．

リクエストの種類

- 以下の5つがあるが，最初の2つがよく使われる.

- GET
  - サーバー側にある情報をクライアントが取得(get)する
- POST
  - サーバー側にクライアントが情報を送る(post)
- HEAD
- PUT
- DELETE

リクエストの構造

- 以下の3つのパートからなる.

- リクエスト ライン
  - どんなリクエストを送るかを示した部分.
- メッセージ ヘッダ
  - メッセージ送り元であるクライアントの特性等のデータ群.
  - Hostのデータは必須.
  - 表示可能な言語，ブラウザの種類，受信可能なデータの種類等を含む.
- メッセージ ボディ
  - 送るメッセージの内容，必ず空行が一行いる.
  - GETの場合は空行一行のみ.
  - ヘッダとボディの間には空行がある.

リクエストの例

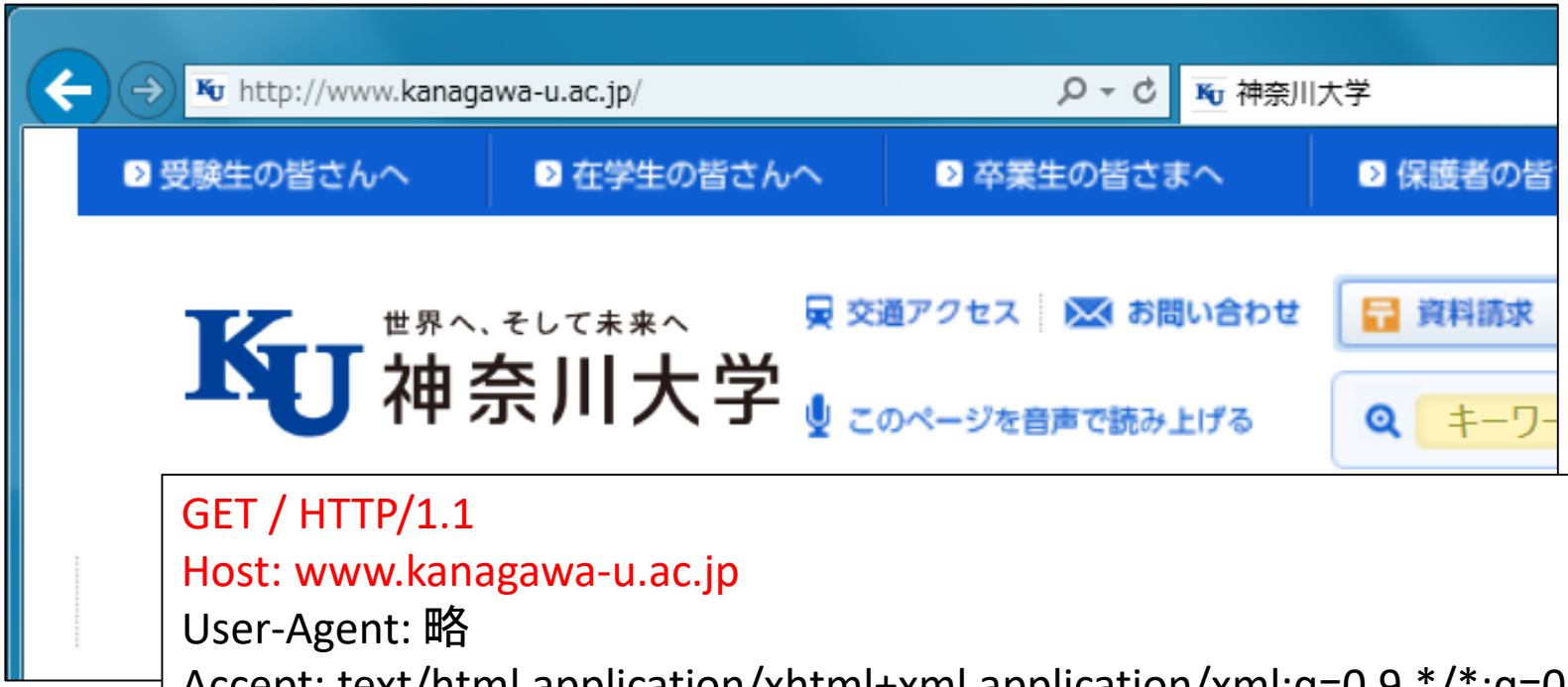
```
GET / HTTP/1.1
Host: www.ana.co.jp
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: ながいので略
Connection: keep-alive
```

全日空のサイト

```
GET / HTTP/1.1
Host: www.kanagawa-u.ac.jp
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:32.0) Gecko/20100101 Firefox/32.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www0.info.kanagawa-u.ac.jp/~kaiya/
Cookie: 長いので略
Connection: keep-alive
Cache-Control: max-age=0
```

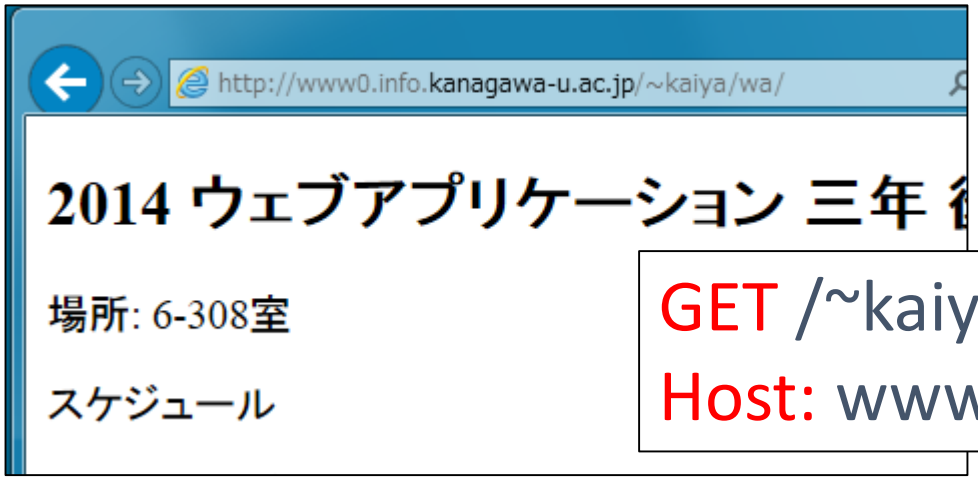
神大のサイト

ブラウザ上の表示との対応



GET / HTTP/1.1
Host: www.kanagawa-u.ac.jp
User-Agent: 略
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://www0.info.kanagawa-u.ac.jp/~kaiya/
Cookie: 長いので略
Connection: keep-alive

リクエストライン等の構造



GET /~kaiya/wa/ HTTP/1.1
Host: www0.info.kanagawa-u.ac.jp

GET /webstation/index.html HTTP/1.1
Host: www.kanagawa-u.ac.jp

レスポンスの構造

- 以下の3つのパートからなる

- ステータス ライン
  - リクエストが成功したか否かの情報.
- メッセージ ヘッダー
  - サーバー側の情報等
- メッセージ ボディ
  - レスポンスの中身. GETの場合, HTMLのページデータが入る.
    - HTMLだけでなく, プログラムも封入される場合がある.
  - ボディとヘッダーの間には空行があり.

レスポンスの例

```
HTTP/1.1 200 OK
Date: Mon, 15 Sep 2014 12:00:06 GMT
Server: Apache/2.2.15 (Scientific Linux)
Last-Modified: Thu, 28 Aug 2014 23:46:13 GMT
Etag: "a1b81-686d-501b924552e21"
Accept-Ranges: bytes
Content-Length: 26733
Connection: close
Content-Type: text/html

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

以下，長いので略
```

神大のサイト

レスポンスの結果の種類

- 200 リクエストが成功した
- 304 ページは更新されてない
- 400 リクエストが不正
- 403 アクセスが禁止されている
- 404 そんなページは無い

サーバーにデータを送付する

- POST等に代表されるリクエストとともにサーバーにデータを送付する方法については，中盤以降に解説します．
- とりあえずは，GETに代表されるページデータを取得する仕組みを理解しておいてください．

- なお，ブラウザでなくても，TCPを理解するプログラムであれば，直接，リクエストをウエブサーバーに送り，ページデータを取得することができる．
  - たとえば，telnet コマンド等．
  - wgetコマンドも，そのように作られている．

通信傍受

- 「HTTP キャプチャ ツール」あたりでググると，HTTPによる通信を傍受をキャプチャするツールがたくさん紹介される．
  - Wireshark とか．
- より一般的に，TCP/IPのキャプチャツールもある．
- 「パケット キャプチャ」

- 紹介のみにとどめますが，関心のある方は試用してみてください．

基盤技術の分類

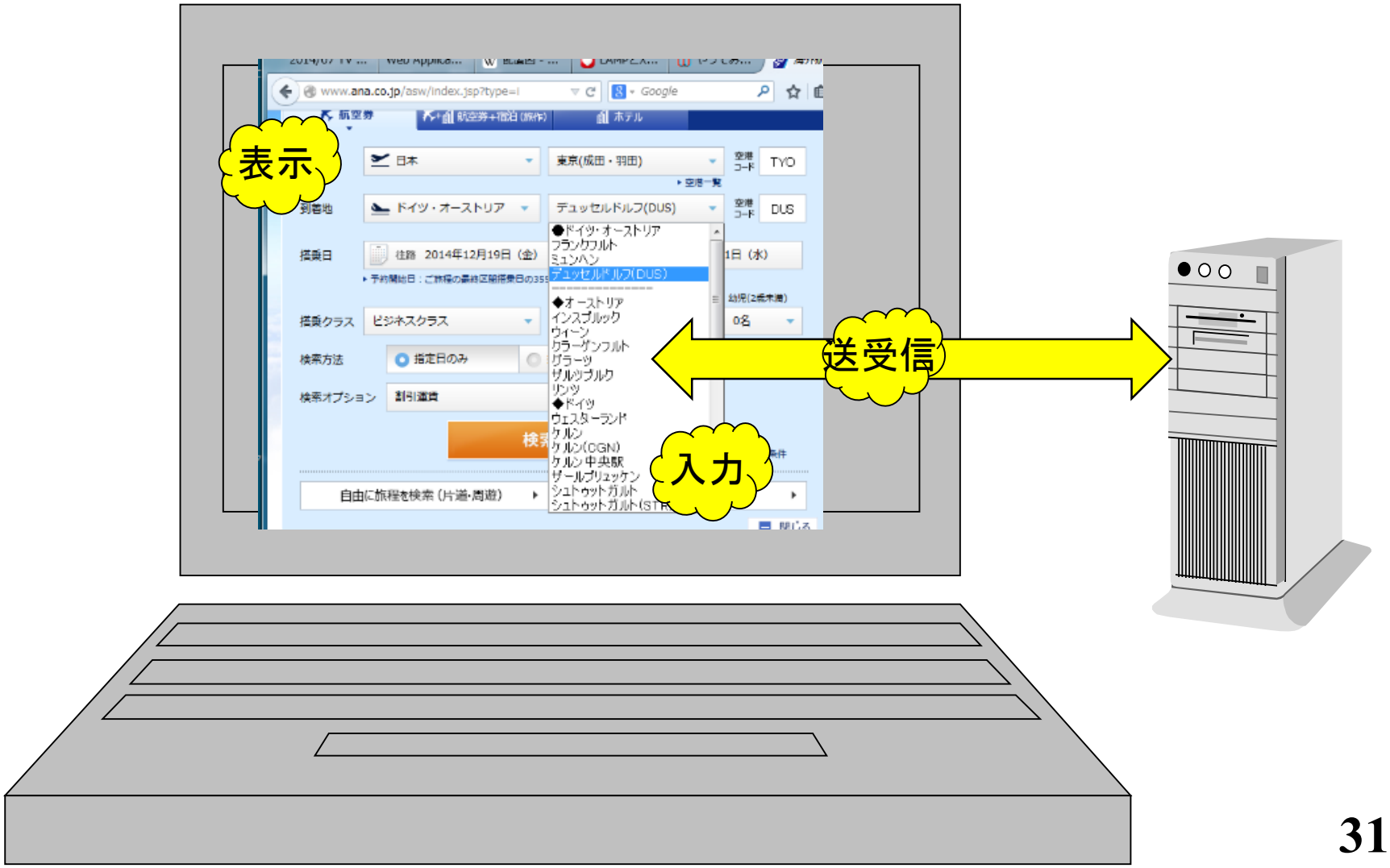
- クライアントサイド・コンピューティング
  - ブラウザ側で処理を行う技術
  - 描画だけでなく，計算も行える.
  - プログラムがネットワーク上で転送される
    - Mobile Code とも呼ばれる.
  - 「フロントエンド」とも呼ぶ.
- サーバーサイド・コンピューティング
  - サーバー側で処理を行う技術.
  - 入出力データがネットワーク上で転送される
    - クライアントとはHTTPで通信する
  - サーバーは一台とは限らない.
    - 例: 商品の発注と，支払いの決済は別のサーバーが行う場合がある.

クライアントサイドの具体的技術

- HTML5, CSS3
  - ブラウザ上で文字や図形を描画するための言語
  - 音声や動画の再生も可能.
  - クライアント側でのデータ保存に関する仕様もある.
- JavaScript
  - ブラウザ上で計算を行うための言語.
  - 基本，C言語に似てる.
  - 文字列やデータ等の変換も計算として行える.
  - 事実上，唯一のクライアントサイドの処理言語
- 廃れたもの
  - Flash, Java/Applet, Java/FX等のRIA (Rich Internet Application)

クライアントサイドの主な役割

- 入力，表示，データの送受信

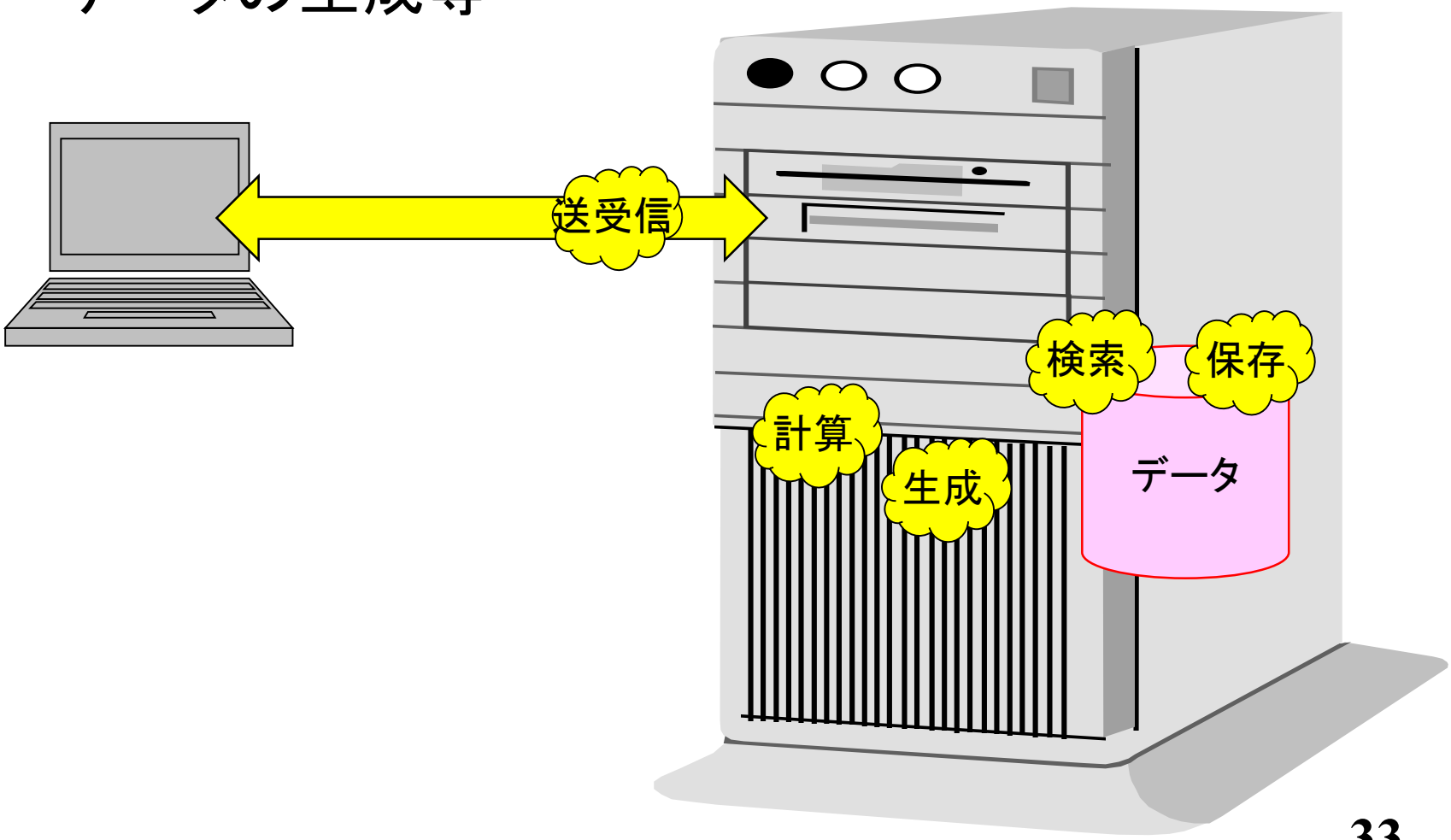


サーバーサイドの具体的技術

- JavaによるJSP/Servlet
  - 銀行系等の大規模で信頼性の必要なシステム向け.
- PHP
  - お手軽にサイトを作るサーバーサイド言語.
  - Cに似ている.
- Ruby on Rails
  - 昨今，流行っている模様.
  - いわゆるモダンフレームワークの走り
- JavaScript
  - Node.js
- 伝統的なCGI (Common Gateway Interface)
  - Perl, Ruby, C等でサーバー側処理をするもの.
  - いまどきは見かけない.

サーバーサイドの主な役割

- 検索，計算，データ保存，手順のナビゲート，画面データの生成等



データの保持

- 業務や娯楽等を継続的に利用するため，クライアントサイド，サーバーサイドそれぞれにデータを保持する機能がある.
- クライアントサイド
  - クッキー
  - ウエブストレージ
- サーバーサイド
  - サーバーOSのファイルにデータを直に読み書き
    - いまどきは，あまり行わない.
  - データベースを利用

データ保持の例

- 一連の処理(セッション)の識別
- 過去の行動を保持
  - たとえば閲覧した商品のリスト等

- 作業途中のデータ保存
  - 例えばゲームにおける艦隊構成や武装
- 作業履歴
  - 購入した商品のリスト
- 個人に紐付けされたデータの保持
  - クレジットカード番号，ポイントの残高等

システムの構成

- 必要なハードウェア群を用意し，その上で必要なソフトウェア群を稼動させる.
- ウエブアプリケーションとしては，最低限，ブラウザとウエブサーバーが必要.
  - ブラウザは利用者が用意する.
  - サーバー群はサービス提供者が用意する.

- どのような処理をどのハード上のどのソフトで行うかも決めなければならない.
  - ソフトウェア・アーキテクチャ設計と呼ばれる.

ハードウェアの種類

- クライアント
  - ウエブブラウザが動作するPC, タブレット，スマートフォン等
  - 一般利用者が直接利用する端末装置となる.
- サーバー
  - サービスを提供するソフトウェアが動作するマシン，Linux, UNIX, Windows Server 等のOSが動作.
  - 複数台のサーバーが連携して動作することも普通.
  - 形態は色々だが，ラックマウントが一般的か.
  - 昨今は仮想化ソフトウェア(VMWare等)上の仮想マシンの場合も多い.

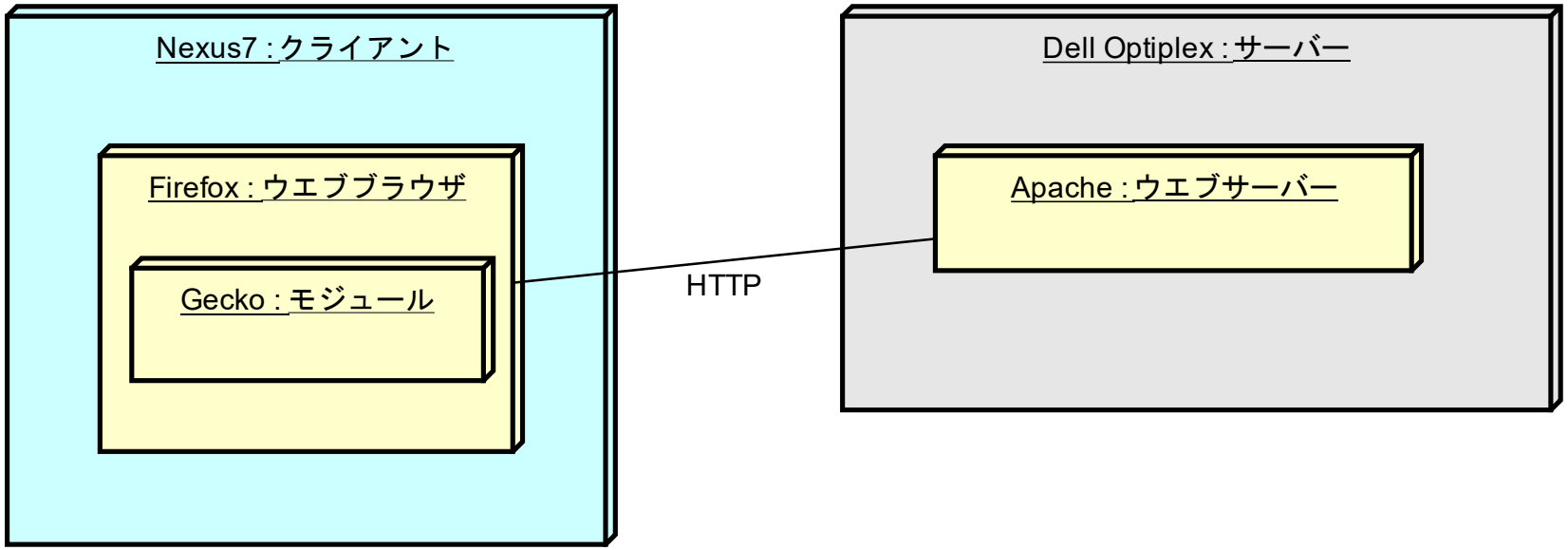
ソフトウェアの種類

- ウエブブラウザ
  - Edge, Firefox, Chrome 等のブラウザ.
- ウエブサーバー
  - HTTPによりブラウザとの通信を行うソフトウェア
  - Apache, IIS, Nginx等が具体例.
- アプリケーションサーバー
  - 特定の業務や活動の手順をガイドするソフトウェア.
  - 本授業ではtomcat(+spring等)がこの位置づけに近い.
- データベースマネージメントシステム (略してデータベース or DBMS)
  - データを永続化するためのソフトウェア.
  - MySQL, MongoDB等
- モジュール
  - 各ソフトウェアの機能拡張をするための部品

※ 実際はOSの仲介がハードとの間に必ず入るが, それは省略.

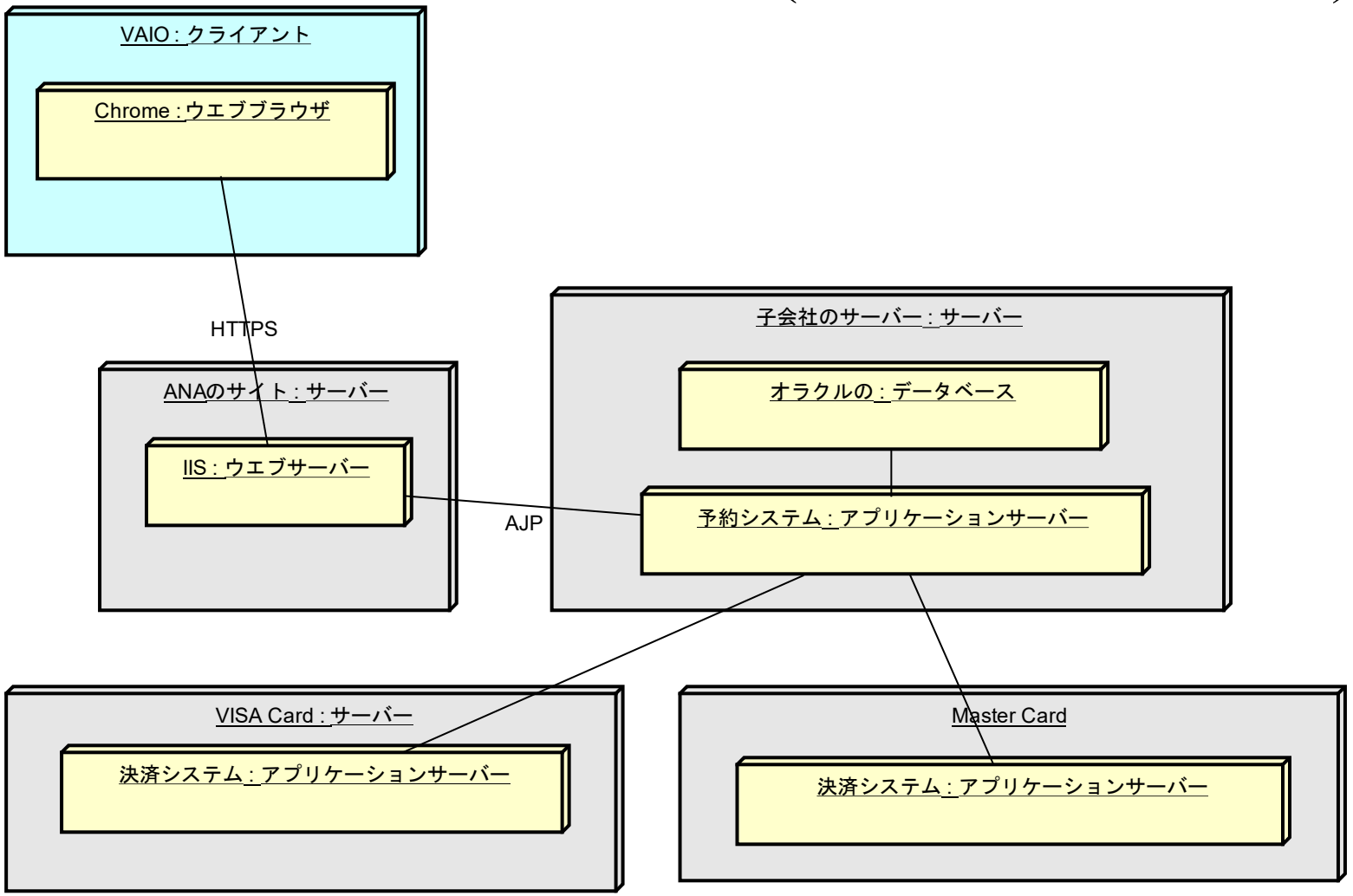
構成例1: 小規模なシステム

- 最低限の体裁をもつウエブアプリケーション



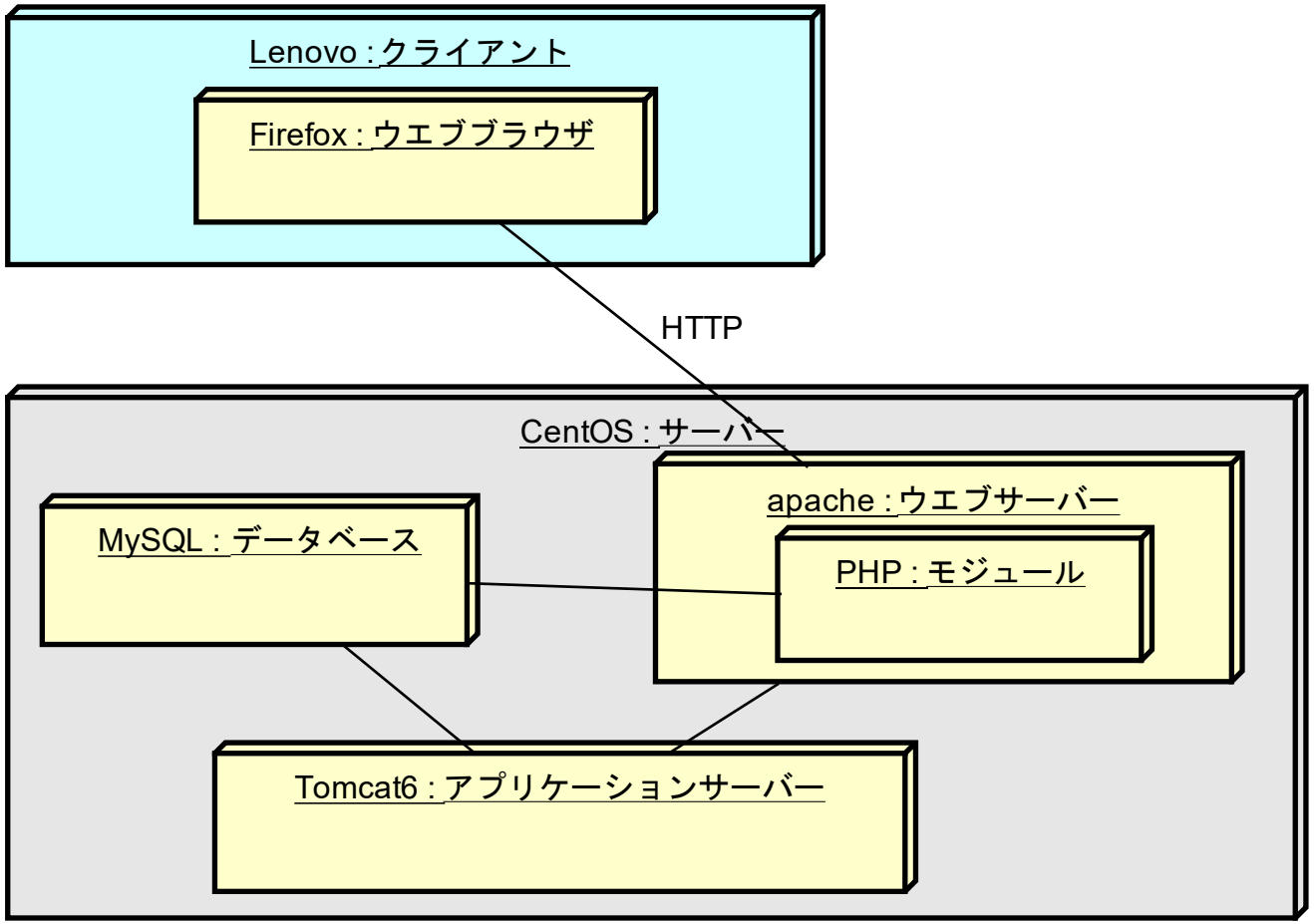
構成例2: 比較的大規模

- 航空券予約等のシステム (現実とは異なります)



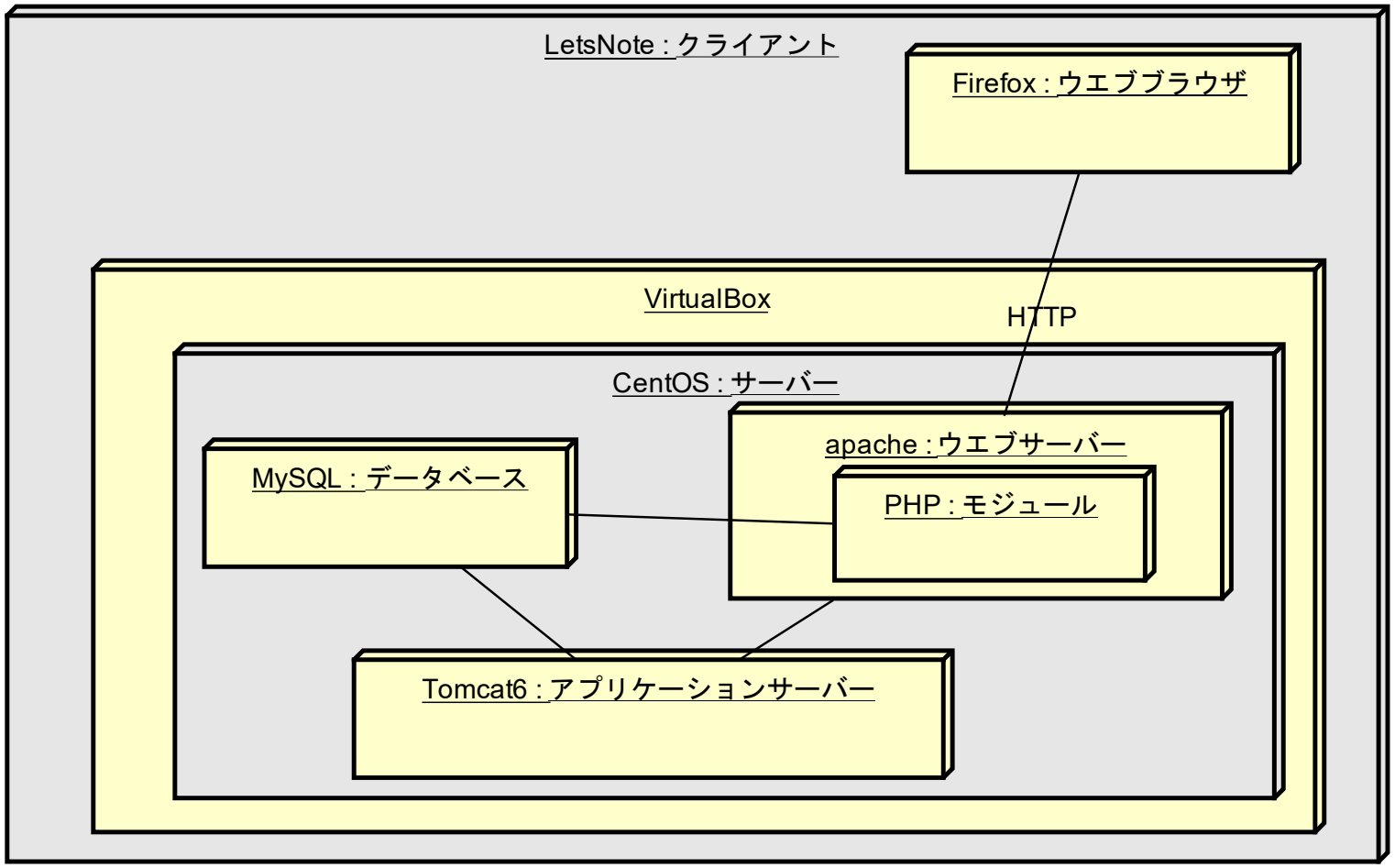
構成例3: よくあるサイト

- いわゆる，LAMP + Java/Servlet



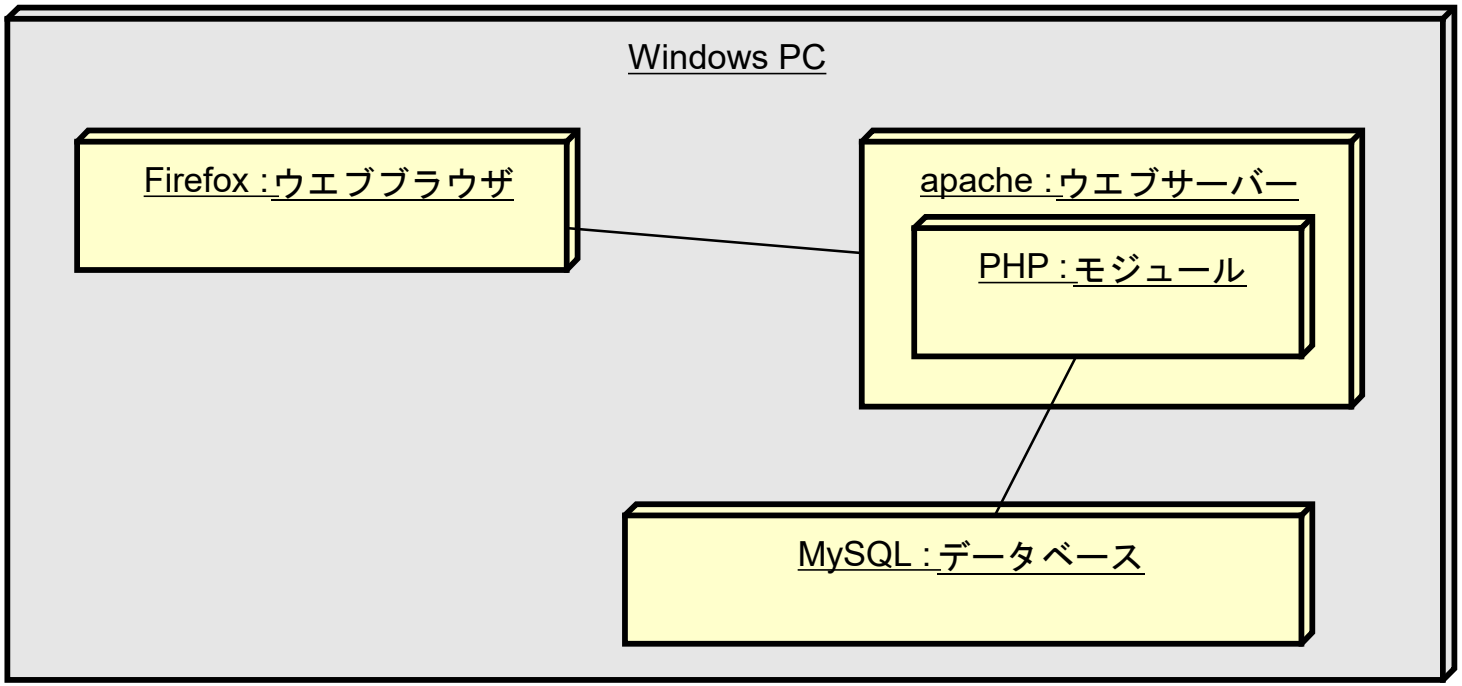
構成例4: 仮想マシンを利用

- 仮想化システムを使い一台のPCで運用.



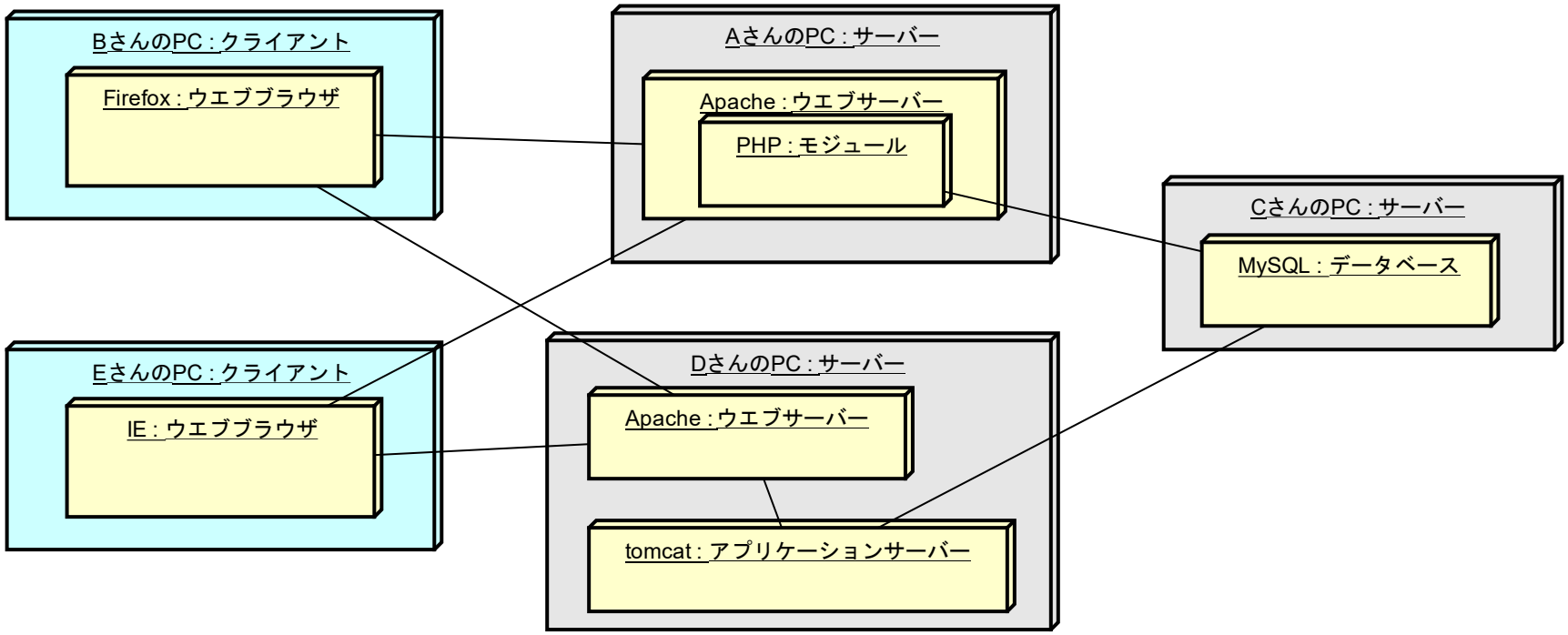
構成例5: 動作テストや開発

- 一台にクライアント・サーバーをやらせる
- いわゆる，XAMPP



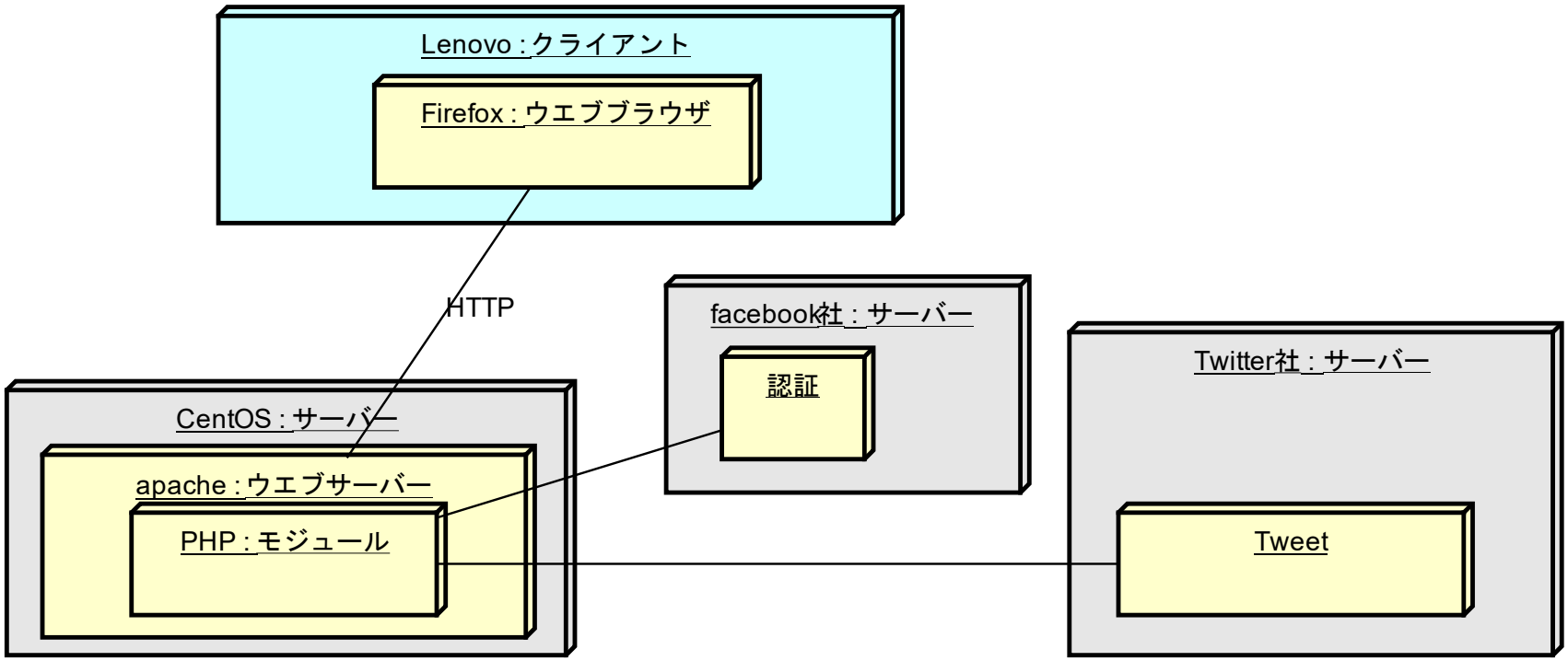
構成例6: 授業中等の実習

- 友達同士でつなぎあってみてください.



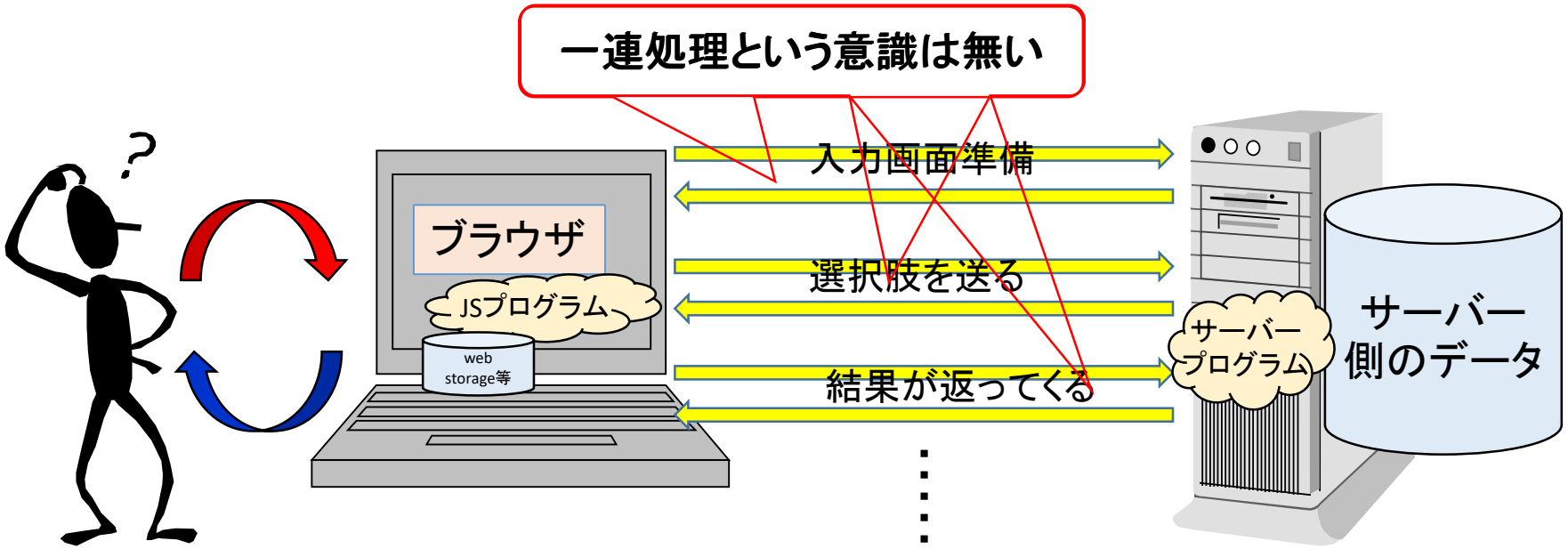
構成例7: SNSの機能を利用

- ユーザー認証や情報公開を外部の他のサーバー(ここではfacebook社とTwitter社)の機能を利用.



ステートレス (状態が無い)

- httpはリクエストとレスポンスの対からなる.
- あるブラウザから連続してリクエスト/レスポンスを行っても，それらの間を関係付けるものは無い.
- このような特徴をステートレスと呼ぶ.

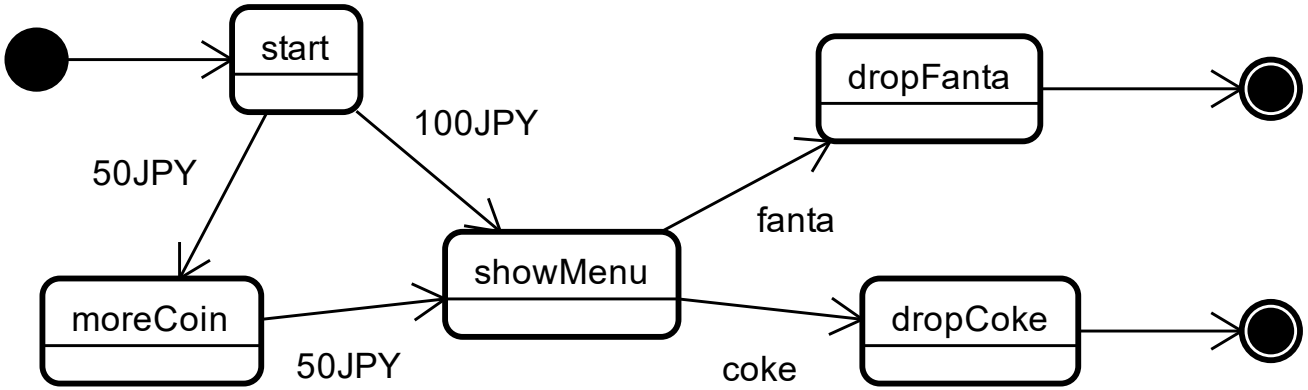


一連のページ遷移を認識

PHPを例題に

ステートレスの例

- 以下のような状態遷移図に従うHTMLページ群を作成することは可能である.
  - サンプル中の sample11/vendor0/ 下のHTMLファイル参照.
- しかし，単なるばらばらのHTMLファイル群なので，状態遷移に沿わないでページにアクセスすることができる.
  - 例えば，金を払わず Coke や Fanta を得られる.



状態をウエブアプリで使うには？

- ブラウザが状態を示す値(状態変数)を記憶し，毎回，サーバーにその値を送信する.
  - ウエブストレージやクッキー(Cookie)が利用可能.

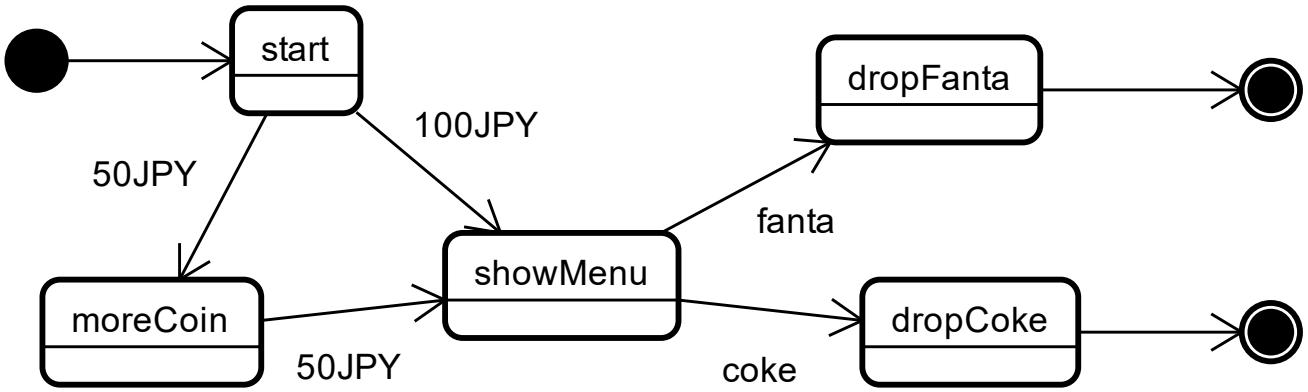
- ブラウザが「一連の処理」を示す識別子を記憶し，毎回，サーバーにその値を送信する.
  - この一連の処理をセッション(session)と呼ぶ.
  - この識別子をセッションIDと呼ぶ.
  - 具体的な状態を示す値は，サーバー側で，セッションIDと対応付けて記録する.

クッキーによる状態継承

- クッキーは Web Storage 同様，クライアントに保持できる小さなkey-value である.
- requestの際に，クライアントからサーバーにクッキーを送ることができる.
  - サーバーが cookie を get できる.
- responseの際に，クライアントにクッキーを設定することができる.
  - サーバーが cookie をクライアントに set できる.
- この情報のやりとりを用いて，状態を連続した処理に継続的に保持することができる.
- 値に日本語は保持できません.

具体例

- サンプルコード中の sample11/vendorC/*.php
- start.php から開始する.
- いきなり showMenu.php 等を実行しようとしても，実行できないようになっている.
  - ページ遷移前の状態名をクッキーに保持しており，下記の遷移から逸脱していると，ページ表示をしないようにしてある.



クッキー

- PHPはクッキーを操作することができる.
- これによって，クライアント側に特定のkey-valueを保持させることにより，状態を維持できる.
- クッキーの設定
  - setcookie関数を利用
- クッキーの取得
  - $_COOKIE連想配列から取り出す.

- 他のサンプル cookie.php

Cookieを用いた問題点

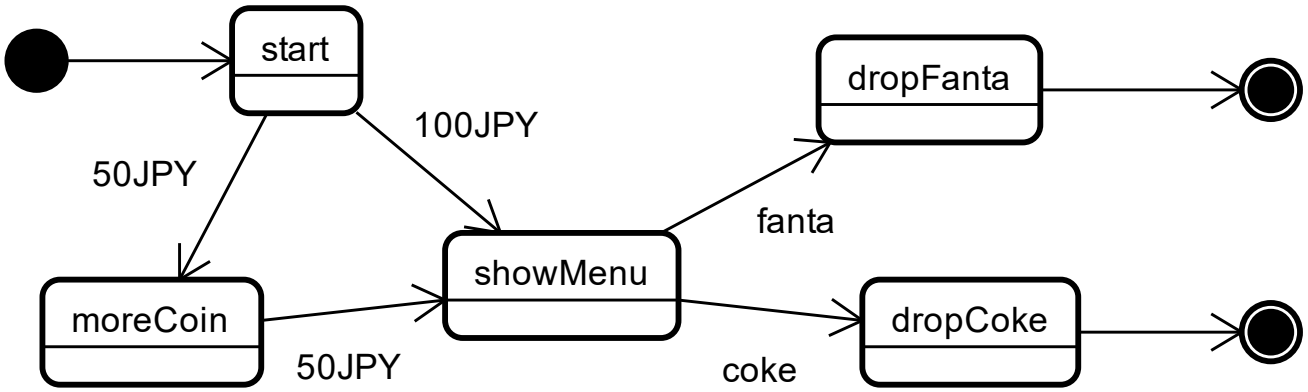
- 状態を表す値そのものがクライアントに保持されるため，改ざんが容易である.
  - 最初の簡単な自動販売機の例なら，状態を表す変数を，利用者が改ざんすれば，遷移を無視した自由が移動ができてしまう.
  - 利用者名やメールアドレス等を簡単に詐称して，なりすましができてしまう.
- 状態を表す値がHTTP リクエストやレスポンスに含まれるため，情報漏えいが容易におきる.
  - HTTPの通信は，とても簡単に傍受できる.

セッション session

- 一連のrequest/response列で状態値群を継承する手段.
- 直接，クライアントのCookieに値群を記録するのではなく，クライアントでは，セッションIDという識別番号のみを記録する.
- クライアントはrequestの際に，このIDを毎回送付する.
- サーバーはこのIDに対応付けられたkey-value(属性名と属性値)を記憶しておく.
- これによって，一連の処理を通して値を継続的に利用することが可能となる.

具体例

- サンプルコード中の sample11/vendorS/*.php
- start.php から開始する.
- いきなり showMenu.php 等を実行しようとしても, 実行できないようになっている.
  - 状態遷移における状態の名前をattributeとして保持して, 以下の遷移からの逸脱を排除している.



セッション

- クッキー同様，PHPでもセッションが実装されている.
- セッション作成 session_start() 関数
  - 作成だけでなく，$_SESSION への値の更新も含む模様.
  - よって，start ページじゃなくてもコレを呼ぶ.
- セッション破棄 session_destroy() 関数
- 変数登録 $_SESSION 連想配列に登録

- 他のサンプル sessionLogin.php

セッションIDの寿命

- どこまでが一連の処理かを明示するため，このIDを破棄するメソッドが存在する.
  - sesson_destory()
  - 1/100の確率で削除されるという謎仕様
- セッションに対しては寿命を設定できる.
  - 長いこと次の処理を行わないと，一連の処理をみなされなくなる.
- デフォルトでは24分間隔があくと寿命がつきるが，この数値も設定変更できる.
- 詳細はAPIマニュアルと例題を見てください.

セッションID

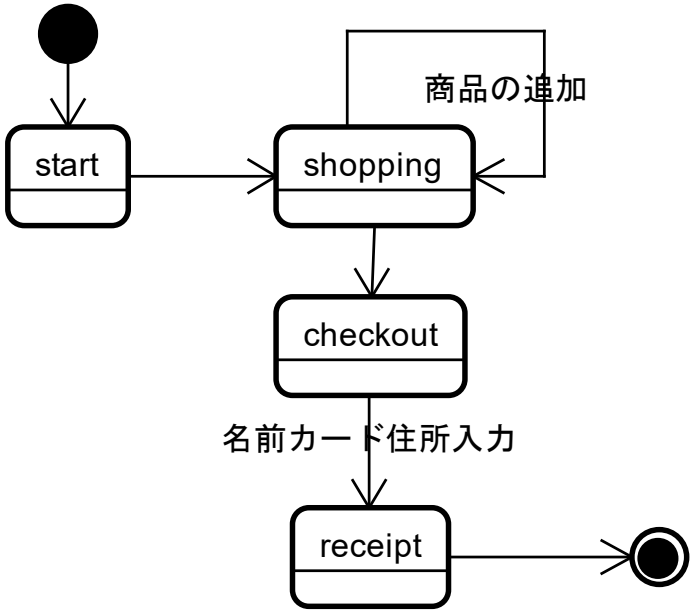
- セッションID自体は，クッキーとして，ブラウザに記録されている.
- クッキーの変数名は PHPSESSID
- この変数を直接使うことは無いので，まぁ，参考程度.
- 実は session_destroy ではこのクッキーは消されないので，確実に消したいなら，

```
setcookie("PHPSESSID", '', time() - 1800, '/');
```

- 等が必要.

他の例題

- サンプルコード中の eshop/ を参照.



- 簡単なオンライン買い物サイト
  - ビール限定
- ちゃんと購入物の累積情報がページ間で継承されている.
- sessionのattributeとして連想配列を使ってみた.

ブラウザにおけるCookie設定

- ブラウザではCookie を受け取らない設定にすることができる.
- コレをすると，session も使えなくなるので，注意が必要.
- 設定法については，個々のブラウザについて調べてください.

Webの歴史

- 1989年 物理学の研究所で便利な道具として発明される.
- 1993年 今のグラフィカルなブラウザの先祖(モザイク)が開発される.
- 1994年 PHPの開発がはじまる.
- 1995年 JavaScriptが開発がはじまる.
- 1997年 Java/Servletの開発はじまる.
- 1999年 Aapacheウエブサーバーの開発はじまる.
- 2008年 HTML5の開発がはじまる.

まとめ

- ウエブアプリケーションとは何か，何が便利なのかの概要を説明しました.
- 通信の要となるHTTPについて説明しました.
- 要となる主たる言語とデータ保持の仕組みを紹介しました.
- システム構成を紹介しました.

本日は以上