1.Webアプリケーション

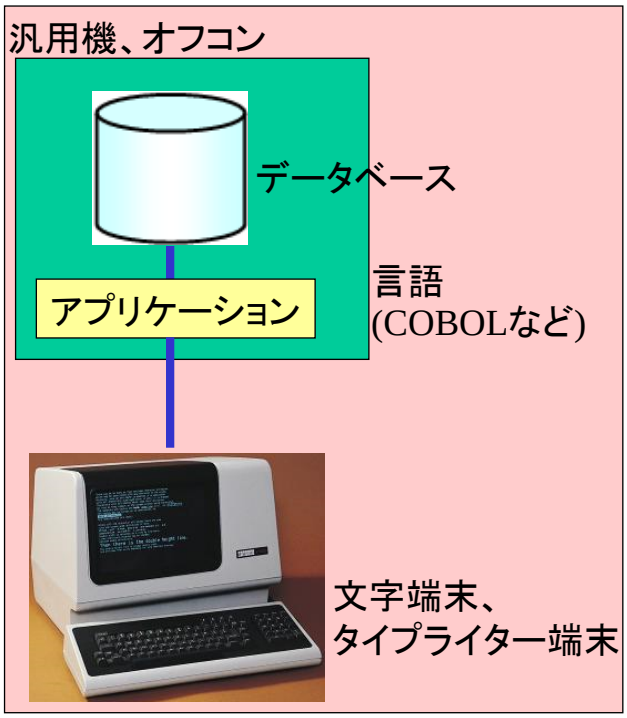
1-1 Webの歴史

- １６年前に誕生
  - 1989年: Tim Berners-LeeがＷＷＷを発明)
  - 1993年: WebブラウザMosaic
  - 1999年: iモード、Windows98
  - 2005年: ？？？

http://www.w3.org/People/Berners-Lee/

1-2 その昔(汎用機の時代)

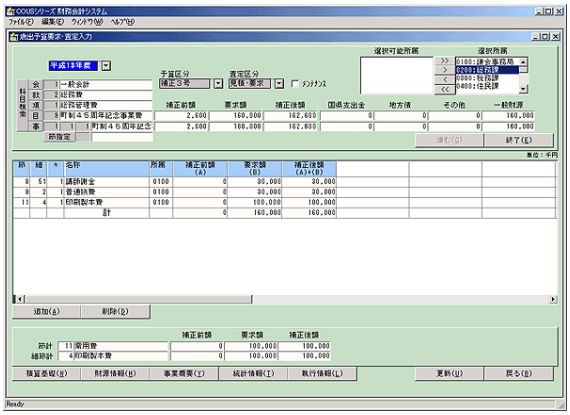
- 高価な計算機を共有
  - 数億円？
  - タイムシェアリング、バッチ処理、課金

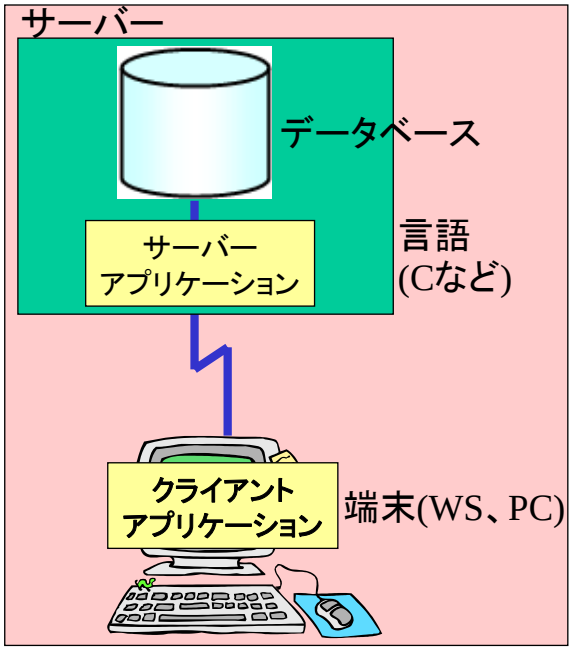


http://research.microsoft.com/~gbell/Digital/timeline/dechistory.htm

1-3 クライアント・サーバー

- 端末でプログラムを実行
  - 安価で高性能な端末
  - インストールが手間、端末の機種を選ぶ
  - イントラネットのみ

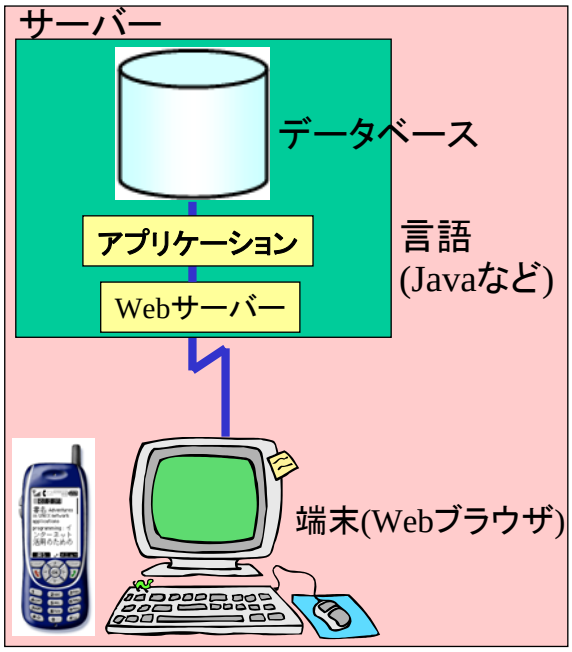




http://www.icc.co.jp/seihin/zaimu/zai1.html

1-4 Webアプリケーション

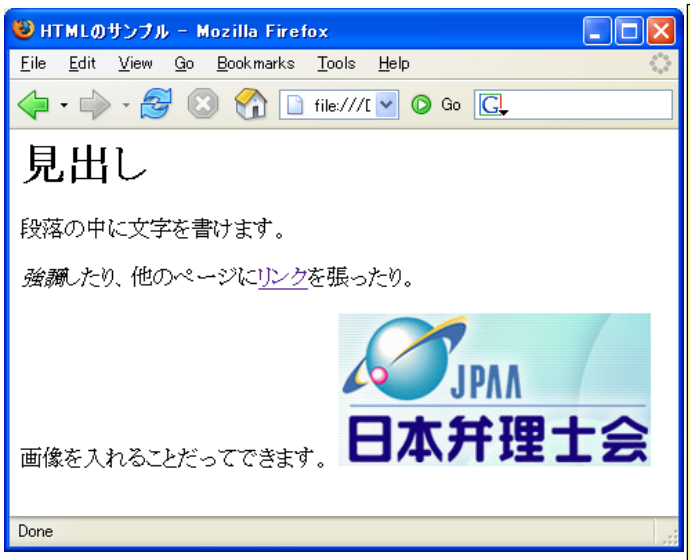
- Webブラウザに表示
  - インストール不要
  - 端末を選ばない、インターネットに対応
    →巨大な市場



2.Webの仕組み

2-1 HTML

- HTML: Hyper Text Markup Language
  - Webページを記述する言語
    (Webページの実体はHTML)




```
<html>
<head>
<title>HTMLのサンプル</title>
</head>
<body>
<h1>見出し</h1>
<p>段落の中に文字を書けます。</p>
<p><em>強調</em>したり、
他のページに
<a href="http://kanemune.cc.hit-u.ac.jp/">リンク</a>
を張ったり。</p>
<p>画像を入れることだってできます。
<img src="jpaa.png">
</p>
</body>
</html>
```

2-2 Webブラウザ

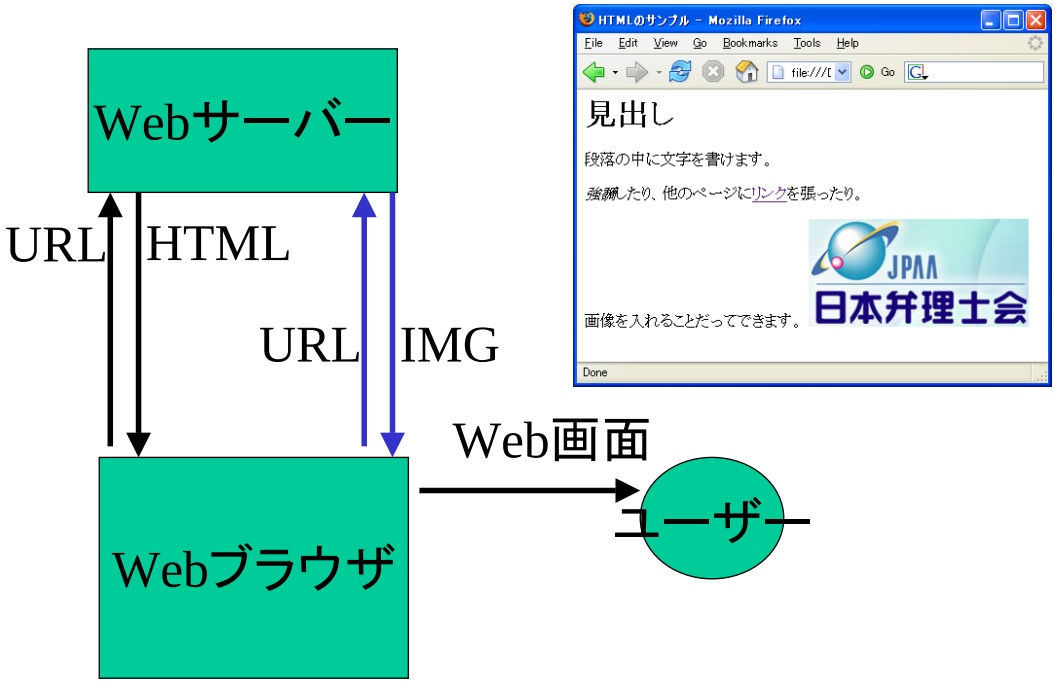
- HTMLを表示するプログラム
  - サーバーからHTMLを受け取り表示する
  - ユーザーの指示をサーバーに送る

```
<html>
<head>
<title>HTMLのサンプル</title>
</head>
<body>
<h1>見出し</h1>
<p>段落の中に文字を書けます。</p>
<p><em>強調</em>したり、
他のページに
<a href="http://kanemune.cc.hit-u.ac.jp/">リンク</a>
を張ったり。</p>
<p>画像を入れることだってできます。
<img src="jpaa.png">
</p>
</body>
</html>
```

2-3 URL

- URL: Uniform Resource Locator
  - インターネット上のデータを示すアドレス
  - 郵便の住所と同様に世界中で一意
  - 書き方
    http://ドメイン/～
  - 例
    http://xxx.co.jp/www/index.html
    http: 通信の種類
    xxx.co.jp: ドメイン(サーバーを示す)
    www/index.html: サーバー内のページ

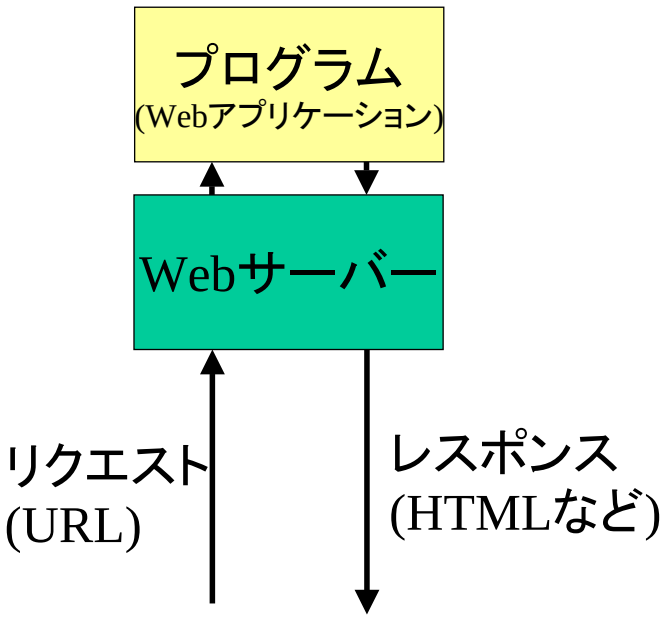
2-4 http通信



3.コンテンツの動的生成

3-1 動的なWebページ

- ＨＴＭＬを「作って」送る
  - 裏でプログラムが動く
    (例)ユーザーに適した広告を送る



3-2 リクエストと引数

- クライアントからデータを送る
  - サーバーのプログラムにデータを渡す

Name: 兼宗 進
Yomi: カネムネ ススム
Addr: 東京都
Sex: male
Submit: OK

3-3 HTTPヘッダ

- ブラウザからWebサーバーに送られる情報

UNIQUE_ID FRoUZoUuChIAAFptYjgAAAAQ

HTTP_HOST kanemune.cc.hit-u.ac.jp

HTTP_USER_AGENT Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 Firefox/1.0

HTTP_ACCEPT text/xml,application/xml,application/xhtml+xml, text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5

HTTP_ACCEPT_LANGUAGE en-us,en;q=0.5

HTTP_ACCEPT_ENCODING gzip,deflate

HTTP_ACCEPT_CHARSET ISO-8859-1,utf-8;q=0.7,*;q=0.7

HTTP_KEEP_ALIVE 300

HTTP_CONNECTION keep-alive

HTTP_REFERER http://kanemune.cc.hit-u.ac.jp/tmp/www/printvars.html

PATH /sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin

SERVER_SIGNATURE Apache/2.0.50 (Fedora) Server at kanemune.cc.hit-u.ac.jp Port 80

SERVER_SOFTWARE Apache/2.0.50 (Fedora)

SERVER_NAME kanemune.cc.hit-u.ac.jp

SERVER_ADDR 133.46.10.18

SERVER_PORT 80

REMOTE_ADDR 203.165.228.64

DOCUMENT_ROOT /var/www/html

SERVER_ADMIN root@localhost

SCRIPT_FILENAME /var/www/html/tmp/www/printvars.php

REMOTE_PORT 52853

GATEWAY_INTERFACE CGI/1.1

SERVER_PROTOCOL HTTP/1.1

REQUEST_METHOD GET

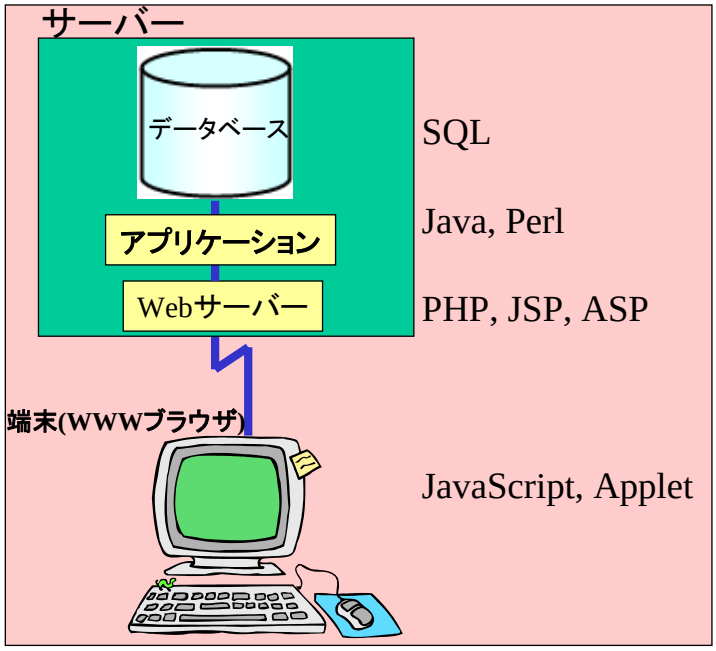
QUERY_STRING food=curry&sex=unknown&cmt=Hello&pw=passwd

REQUEST_URI /tmp/www/printvars.php?food=curry&sex=unknown&cmt=Hello&pw=passwd

4.Webアプリケーションで使われる言語

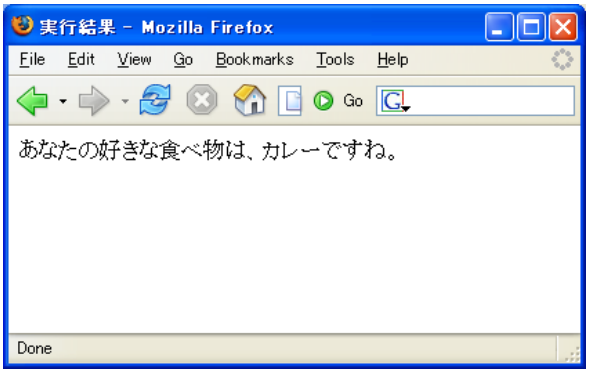
4.1 言語の動く場所

- Webアプリケーション関連のプログラム
  - さまざまな場所で動く



4-2 HTML埋め込み言語の例(PHP)

- HTMLの中にプログラムを埋める
  - 簡単!




```
<html>
<head>
<title>実行結果</title>
</head>
<body>
あなたの好きな食べ物は、
<?php print($_POST["food"]); ?>
ですね。
</body>
</html>
```

4-3 クライアント言語

- Webブラウザ上で動く
- HTMLに埋め込まれる
  - JavaScript: HTML処理を補助
- サーバーからプログラムを転送して実行
  - JavaApplet: 携帯アプリなど
  - ActiveX: Windowsプログラム
  - Flash: 画面でアニメーションなど
  - ...

4-4 クライアント言語の例(JavaScript)

- JavaScript
  - HTML中に記述
  - 通信しない
  - 送信前の処理など

```
<html><head>
<title>Formサンプル</title>
<meta http-equiv="Content-script-type" content="text/javascript">
<script type="text/javascript">
function check() {
  var f1 = document.forms[0].elements[0].checked;
  var f2 = document.forms[0].elements[1].checked;
  var f3 = document.forms[0].elements[2].checked;
  if (!f1 && !f2 && !f3) {
    window.alert("どれかをチェックしてください"); return false;
  }
  return true;
}
</script>
</head><body>
<form method="POST" action="form1.php" onsubmit="return check()">
好きな食べ物:
<input type="checkbox" name="food" value="ハンバーグ" />ハンバーグ
<input type="checkbox" name="food" value="ラーメン" />ラーメン
<input type="checkbox" name="food" value="カレー" />カレー
<input type="submit" value="送信" />
</form></body></html>
```

4-5 セッション管理

- http通信: 1画面ごとに通信が切れる
  - CGI: 1画面の実行ごとにプログラムが終る
    → 状態や前の画面の情報を知らない！
- しかし、
  - ユーザー: 買物カゴに入れて買い物中
  - システム: データベースの処理中

→ どうやって状態を管理するか？

4-6 クッキー

- サーバーからWebブラウザに情報を埋める
  - サーバーは後で(自分の埋めた)情報を見る
  - 他のサーバーの情報は見えない
  - サイズや個数は制限されている
  - 有効期限を過ぎたら消える
- 弱点
  - クッキーを受け付けるかはユーザーまかせ
  - ブラウザやPCが変わると他人扱い

4-7 Formの隠し項目とURL引数

- Formのhidden項目
  - formの送信時に同時に送られる(見えない項目)

```
<INPUT type="hidden" name="SHOP_ID" value="BICBIC">
<INPUT type="hidden" name="PRODUCT_NAME" value="オールインワン・プリンタ　HP PSC1315">
<INPUT type="hidden" name="SCREEN_ID" value="bw030000">
<INPUT type="hidden" name="ActionType" value="bw030000_RE">
<INPUT type="hidden" name="PRODUCT_ID" value="0010059639">
<INPUT type="hidden" name="BUY_PRODUCT" value="0010059639,14800">
```

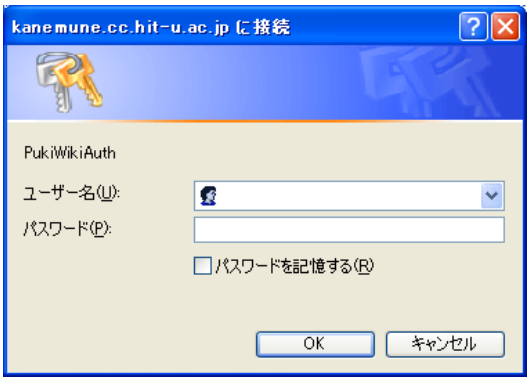
- URL引数
  - URLの末尾にデータを付けて送る

```
http://www.biccamera.com/bicbic/app/w?SCREEN_ID=bw011100&fnc=f&ActionType=bw011100_01&PRODUCT_ID=0010059639&BUY_PRODUCT=0010059639,14800
```

- 特徴
  - 画面から画面へデータを受け渡す
  - form以外のページをはさむと消える。戻るボタンは?

4-8 基本認証

- Webを認証する
  - IDとパスワードでアクセスを保護



(詳しい認証の話は第5回で)

4-9 基本認証の裏側

- 設定ファイル「.htaccess」
  - そのWebディレクトリの情報を記述
  - 以下のように書くとアクセスに認証が必要

```
AuthType Basic
AuthName "IDとパスワードを入れてください"
AuthUserFile /home/susumu/etc/htpasswd
```

- パスワードファイル
  - IDと暗号化されたパスワードを記録する

```
waseda:gY8lIFcT1BBf2
```

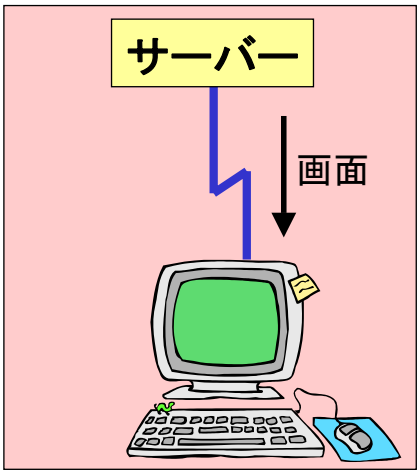
  - 実用システムではDBで管理することが多い

5.Webサービス

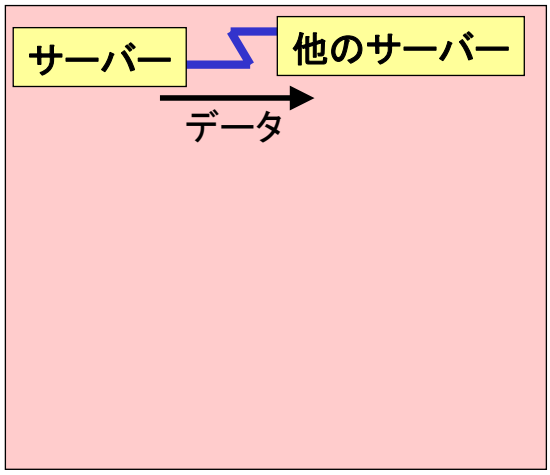
5-1 Webサービス

- 計算機と通信
  - インターネットで取引相手が世界に拡大
  - (×)独自データ→(○)ＸＭＬ

WWW



Webサービス

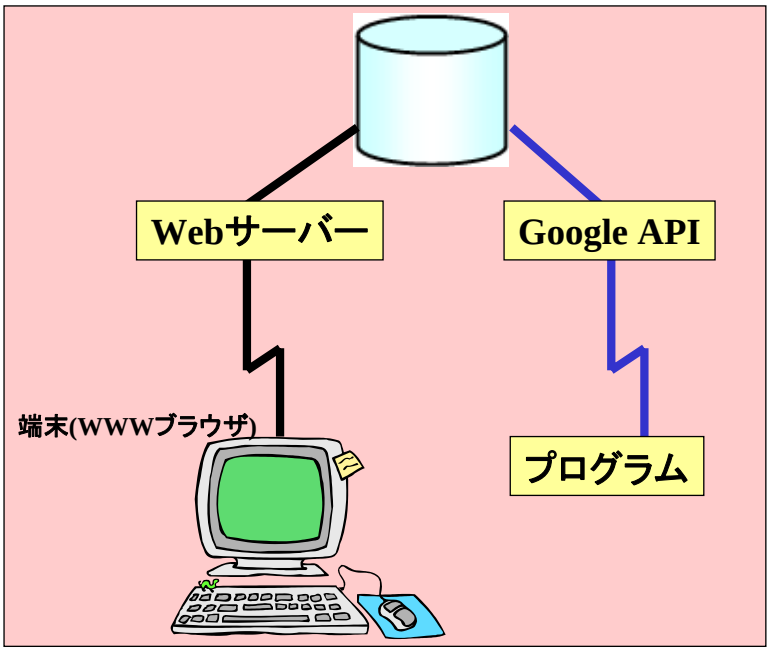


5-2 要素技術

- SOAP
  - 通信。XMLを転送する
- XML
  - データ形式
- WSDL
  - 通信形式の説明
- UDDI
  - ディレクトリサービス
  - 取引先を探す。企業データなどを提供

5-3 GoogleのWebサービス

- Web API
  - プログラムからGoogleにアクセス可能



5-4 実行例

■ JavaプログラムとWSDL定義ファイル

VIM - d:¥win32app¥googleapi¥GoogleAPIDemo.java

File Edit Global Setup Config Prev File Next File Help

```
    // Create a Google Search object, set our authorization key
    GoogleSearch s = new GoogleSearch();
    s.setKey(clientKey);

    // Depending on user input, do search or cache query, then print
out result
    try {
      if (directive.equalsIgnoreCase("search")) {
        s.setQueryString(directiveArg);
        GoogleSearchResult r = s.doSearch();
        System.out.println("Google Search Results:");
        System.out.println("======================");
        System.out.println(r.toString());
      } else if (directive.equalsIgnoreCase("cached")) {
        System.out.println("Cached page:");
        System.out.println("============");
        byte [] cachedBytes = s.doGetCachedPage(directiveArg);
        // Note - this conversion to String should be done with refer
ence
        // to the encoding of the cached page, but we don't do that h
ere.
        String cachedString = new String(cachedBytes);
        System.out.println(cachedString);
      } else if (directive.equalsIgnoreCase("spell")) {
        System.out.println("Spelling suggestion:");
        String suggestion = s.doSpellingSuggestion(directiveArg);
        System.out.println(suggestion);
      } else {
        printUsageAndExit();
      }
```

VIM - d:¥win32app¥googleapi¥GoogleSearch.wsdl

File Edit Global Setup Config Prev File Next File Help

```
  <message name="doSpellingSuggestion">
    <part name="key"          type="xsd:string"/>
    <part name="phrase"       type="xsd:string"/>
  </message>

  <message name="doSpellingSuggestionResponse">
    <part name="return"       type="xsd:string"/>
  </message>

  <!-- note, ie and oe are ignored by server; all traffic is UTF-8. -
->

  <message name="doGoogleSearch">
    <part name="key"          type="xsd:string"/>
    <part name="q"            type="xsd:string"/>
    <part name="start"        type="xsd:int"/>
    <part name="maxResults"   type="xsd:int"/>
    <part name="filter"       type="xsd:boolean"/>
    <part name="restrict"     type="xsd:string"/>
    <part name="safeSearch"   type="xsd:boolean"/>
    <part name="lr"           type="xsd:string"/>
    <part name="ie"           type="xsd:string"/>
    <part name="oe"           type="xsd:string"/>
  </message>

  <message name="doGoogleSearchResponse">
    <part name="return"       type="typens:GoogleSearchResult"/>

  </message>
```

5-5 実行結果

■ JavaからGoogleを検索した結果

```
Google Search Results:
======================
{
TM = 0.037146
Q  = "kanemune"
CT = ""
TT = ""
CATs =
  {
  <EMPTY>
  }
Start Index = 1
End   Index = 10
Estimated Total Results Number = 120
Document Filtering = true
Estimate Correct = false
Rs =
  {

  [
  URL  = "http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/k/Kanemune:Susumu.html"
  Title = "DBLP: Susumu <b>Kanemune</b>"
  Snippet = "Susumu <b>Kanemune</b>. <b>...</b> 1, EE, Susumu <b>Kanemune</b>, Takako Nakatani, Rie Mitarai,
Shingo Fukui,<br> Yasushi Kuno: Dolittle - Experiences in Teaching Programming at K12 Schools. <b>...</b>  "
  Directory Category = {SE="", FVN=""}
  Directory Title = ""
  Summary = ""
  Cached Size = "3k"
  Related information present = true
  Host Name = ""
  ],

  [
  URL  = "http://kanemune.cc.hit-u.ac.jp/kanemune/"
  Title = "兼宗研究室 - <b>Kanemune</b> Laboratory, Hitotsubashi University"
  Snippet = "[PukiWiki] 兼宗研究室. http://<b>kanemune</b>.cc.hit-u.ac.jp/<b>kanemune</b>/pukiwiki.<br> php?
B7%F3%BD%A1%B8%A6%B5%E6%BC%BC. [ リロード ] [ 新規 <b>...</b>  "
  Directory Category = {SE="", FVN=""}
  Directory Title = ""
  Summary = ""
  Cached Size = "12k"
  Related information present = true
  Host Name = ""
  ],
```

6.WWW検索

6-1 検索エンジン

- WWW利用の中心
  - 数十億ページから検索～数十万件ヒット～10件表示

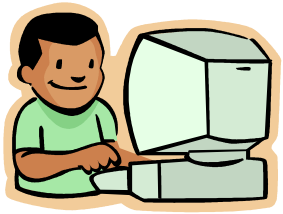
Googleは80億ページ収集

「弁理士」で18万ページヒット

6-2 従来の情報検索との比較

■ 検索の使われ方がまったく異なる

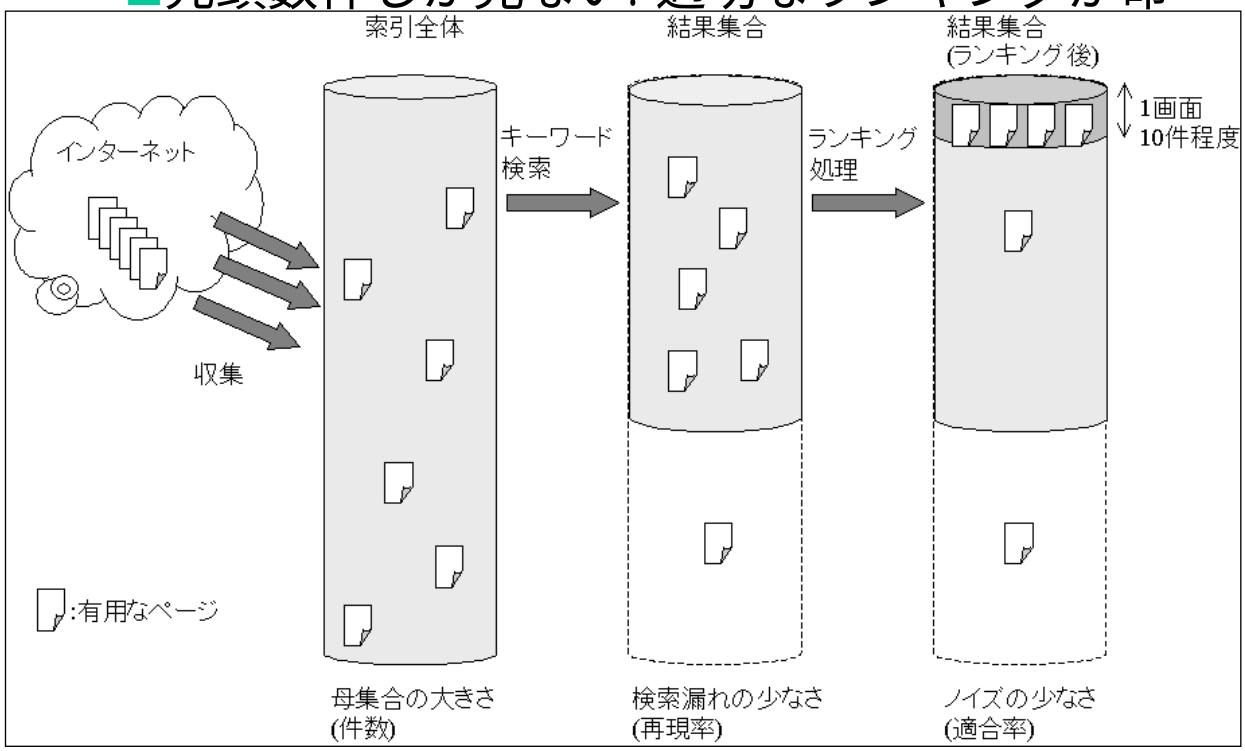
項目	従来の情報検索	検索エンジン
ユーザー	サーチャー	初心者
検索語	吟味した検索式	思い付いた1,2語
結果の閲覧	全件	先頭10件
絞り込み	する	しない
結果集合	数十～数千件	数万～数百万件
求める情報	網羅的	数件



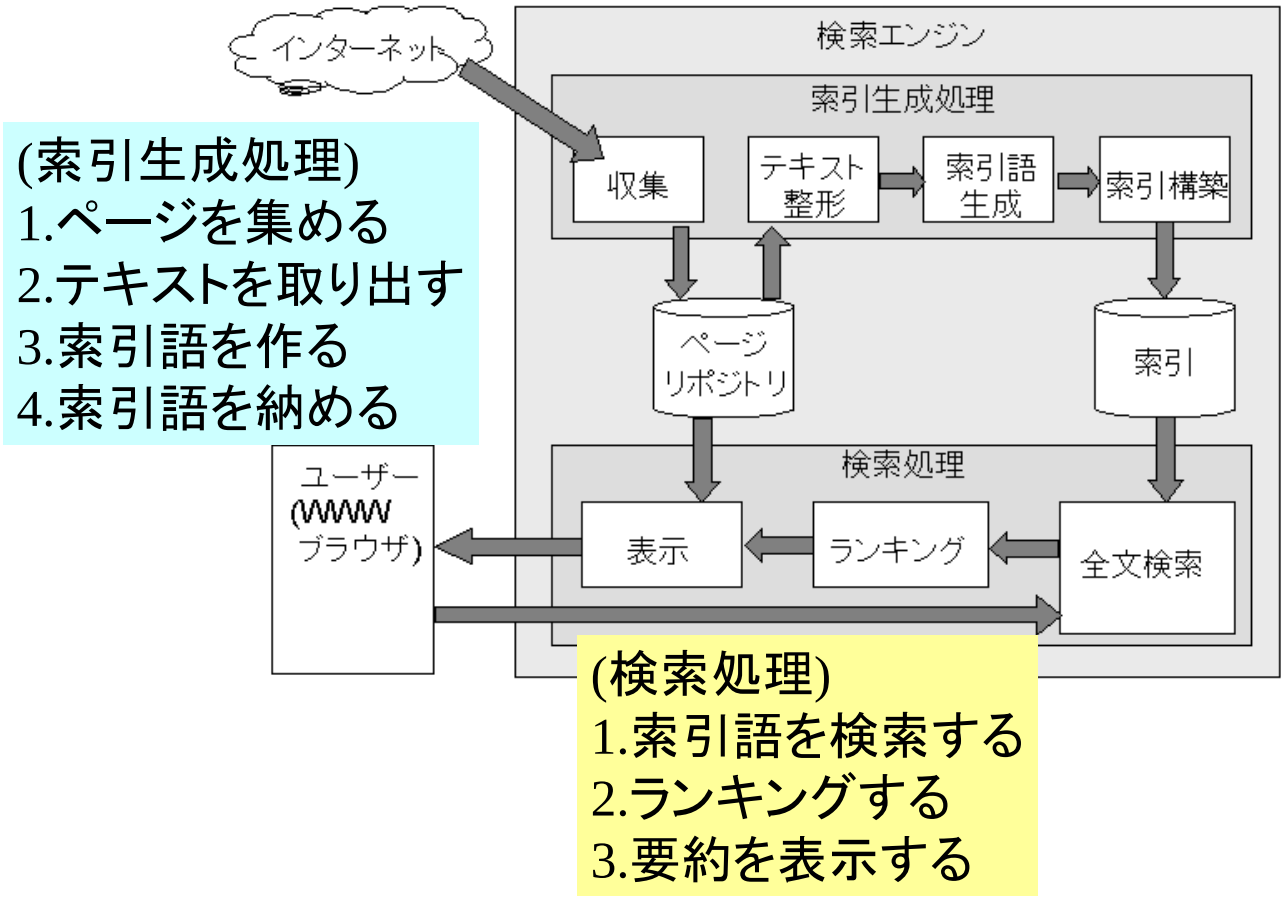
6-3 検索の流れ

■ 結果集合を絞らない

■先頭数件しか見ない: 適切なランキングが命



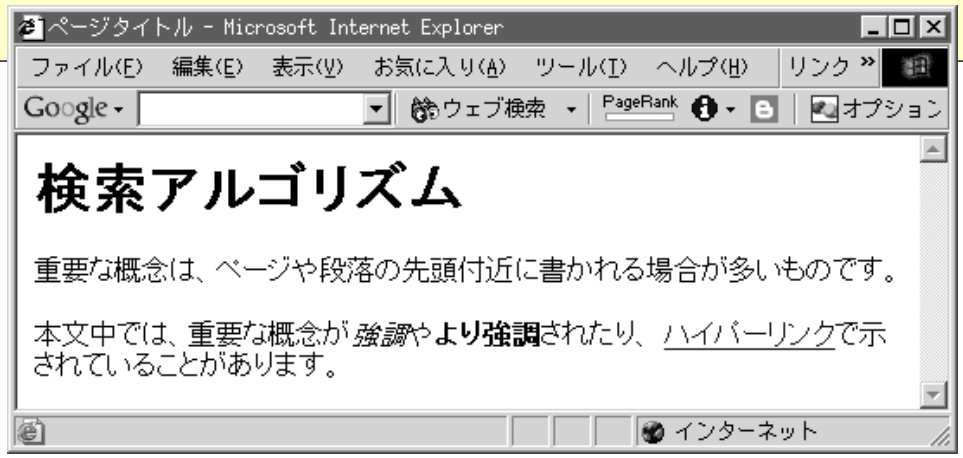
6-4 検索エンジンの構成



6-5 テキストの取り出し

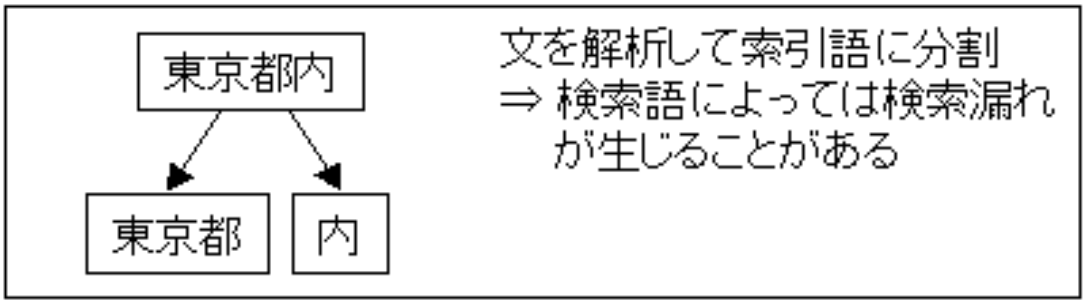
■ 画面に表示される情報が中心

```
<html lang="ja">
<head>
<title>ページタイトル</title>
</head>
<body>
<h1>検索アルゴリズム</h1>
<p>重要な概念は、ページや段落の先頭付近に書かれる場合が多いものです。</p>
<p>本文中では、重要な概念が<em>強調</em>や<strong>より強調</strong>されたり、
<a href="link.html">ハイパーリンク</a>で示されていることがあります。</p>
</body>
</html>
```



6-6 索引語の生成(1)

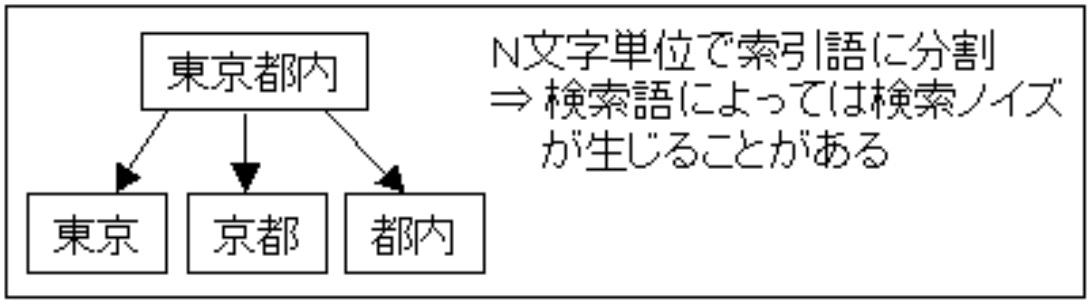
- 日本語: 単語の区切りがない→分割する
- 手法1: 形態素解析(単語ごとの分割)
  - 辞書と解析プログラムで日本語を分割する



【検索例】 東京都 ⇒ 検索される
京都 ⇒ 検索されない
都内 ⇒ 検索されない(検索漏れ)

6-7 索引語の生成(2)

- 手法2: N-gram(文字ごとの分割)
  - 隣接するN文字で分割する



【検索例】 東京都 ⇒ 検索される
京都 ⇒ 検索される(検索ノイズ)
都内 ⇒ 検索される

6-8 ランキング

- よいページを先頭に表示できるかが勝負
- 「よいページ」とは?
  - さまざまな手法。検索エンジンごとの企業秘密
  - ページに点数を付ける(スコアリング)
  - 3種類に分類してみた

基になる情報	スコアリング手法	類似の概念
データの特性	出現頻度、タグ、出現位置、近接度	カーナビゲーション、鉄道経路検索
ユーザーの行動	クリック人気	ベストセラー情報
ユーザーの推薦	リンクポピュラリティ	文献の引用情報

6-9 検索エンジンの限界とRSS

- 検索エンジンの索引更新
  - 巡回して索引に反映
    - ➡数十億ページの短期間での巡回は不可能
    - ➡巡回頻度に重みを付けて対応(巡回戦略)
  - 数日から数週間→遅すぎる!
- 新しいニーズの出現

ページの種類	ディレクトリ	検索エンジン	RSS
更新されない	◎	○	?
週単位の更新(A)	×	◎	?
分単位の更新(B)	×	×	○
DBから作成(C)	×	×	×
他からリンクされない	×	×	×

まとめ

- こんなことを学びました
  - Webアプリケーションの仕組み
  - 使われる言語
  - 事例1: サーバー間通信(Webサービス)
  - 事例2: 情報検索