情報システムとWeb技術＃1 — 情報システムとネットワーク

久野　靖*

2011.10.4

0　はじめに

本科目「情報システムとWeb技術」は、今日の世の中の基盤(インフラストラクチャ)となっている情報システムについて、それがどのようなものであり、どのようにして動いているのかについて、使用者側からの視点だけでなく、発注者・設計者・構築者の視点まで含めて、概要を理解することを目的としています。といっても、このような遠大なテーマをたった5回で詳しくカバーすることなど到底無理ですから、本科目では次のような方針を採用します。

まず、情報システムの(構想・設計・構築の)全体的な考え方については、各回とも重くなりすぎない範囲で側面を絞ってトピック的に取り上げることにします。そしてその残りについては、個別の題材について例題を用いて技術的な側面を中心に見て行きます。具体的な題材としては、現在の情報システムにおいて重要な位置を占めるWebシステムを用い、その実装の個別の側面に絞って取り上げて行くことにします。ここが「…とWeb技術」に相当するわけです。

個別の例題については、実際に例題プログラムを動かしてみながら、それを参考に自分なりに手直ししたり新たなコードを追加したりして、課題をこなして頂くようにします。プログラムにはJava言語を使用していますが、これはこの言語が汎用的なネットワークプログラミングや、情報システム技術によく適合しているからです。Java言語に熟練している必要はありませんが、Java言語のプログラムが「怖くない」程度のスキルは前提とさせて頂きます。

各回において取り上げるテーマとしては、とりあえず次のように予定しておきます(この科目は今年度が最初なので、実際にはじめて見ないと分からないところもありますから、これはあくまでも予定です)。

＃1 情報システムとネットワーク

- 情報システムの定義、情報システム定式化の手法、構造化分析
- ネットワークの位置づけと役割
- 簡単なネットワーク側プログラミング
- 簡単なサーバ側プログラミング
- Webサーバの基本的な考え方と枠組み

＃2 データベースの位置づけと3層モデル

- データモデルとERモデル、データベース設計
- SQLとJDBCによるデータベース操作
- GUIを持つデータベースアプリケーション(2層)
- Webインタフェースを持つデータベースアプリケーション(3層)

＃3 Webサーバとセッション管理

*経営システム科学専攻

- 業務モデルと業務分析
- セキュリティと性能評価
- サーブレット、サーブレットコンテナ、セッション管理
- フォーム入出力

＃ 4 伝統的 Web アプリケーション

- データウェアハウス/情報系データベース
- フォームベースの Web アプリケーション
- SOA と REST
- 総合製作 (1)

＃ 5 Web アプリケーションのユーザエクスペリエンス

- ユーザエクスペリエンスとその評価
- クライアント側スクリプト、Ajax、Canvas
- 総合製作 (2)

本科目の評価については、各回の出席 (議論・実習への参加) プラス最終レポートによるものとします。最終レポートは、各回の資料に含まれているさまざまな課題から好きなものを 1 つ以上選んでレポートを書いて頂くものです。

では、これから 5 回にわたり、よろしくお願いします。

1 情報システムとその定式化

1.1 情報システムの定義とその意義

システム (system、体系) とは、全体としてまとまった機能を実現するような、構成要素の有機的な集まりを意味します。そして情報システムとは、情報を取り扱うことを目的としたシステムを言います。

たとえば人間や昆虫などの生物の場合、情報を扱うことがその働きの大きな部分を占めることは確かだがが、情報を取り扱うことを目的としているかどうかについては疑問があるので、普通は情報システムであるとは言わないでしょう。諜報機関のような組織について考えれば、確かに情報を扱うことを目的としたシステムであるので、上の定義にあてはまります。

しかし今日では、最も広く見られる情報システムは、コンピュータを中心としたシステムということになるでしょう。というのは、コンピュータは情報を取り扱うことを目的とした装置であり、しかもその出現以前には例が無かったほどに (人間の手作業による処理では不可能なほどに)、大量の情報を正確・高速に取り扱うことを可能にしているからです。

人間を他の動物を制して地球の支配者たらしめたものは、人間が持つ情報処理能力であることは間違いありません。そして、人間が持つ情報処理能力を (ごく単純な部分に限られているとはいえ) 代替し、かつ増強することを可能にした情報システムというものが、人間の存在にとって重要なものであることもまた確かです。そもそもそんな理屈を言わなくても、私たちは日常的に多くの情報システムに囲まれ、その恩恵を受けながら生活しています。

質問 1 今日私たちの身のまわりにある情報システムとして、具体的にどのようなものがあるか、それは情報を「どのように」扱っているかをいくつか思い付く範囲で列挙してみなさい。

1.2 情報システムの定式化

では次に、ある情報システムがあったとして、それはどのように定式化できるでしょうか。もっとひらたく言えば、それがどのような構造を持ち、その個々の要素がどのような働きを受け持っているかを、きちんと表現するにはどのようにしたらよいでしょうか。

もちろんそのために、先人がさまざまな手法を考案したり定式化したりしてきています。その中にはZのような論理式に基づく (形式的手法に近い) ものもありますが、ぱっと見分かりやすいのは図を用いた方法でしょう。

図による記法もまた色々な流儀がありますが、ここではデマルコによる階層化されたデータフロー図 (DataFlow Diagram、DFD) を紹介します。デマルコが提唱した手法は「構造化分析 (Stractured Analysis)」と呼ばれ、システムを階層的に分解して「どのようなシステムであるか」を同定する (分析する) ことを目的とした手法です。DFDはそのための記法なのですが、今では図法であるDFDの方が有名になっています。

質問 2 あなたはシステムを記述し分析するのにどのような手法を使っていますか。またはどのような手法を知っていますか。その手法の利点や弱点は何ですか。

1.3 構造化分析入門

構造化分析のDFDでは、○でプロセス (何らかの処理) を表し、矢線でデータの流れを表し、線分でデータの蓄積場所 (ファイルやDB) を表します。構造化分析では基本的にはこの3つと、あとシステムの外部にあるエンティティを表す□だけで、情報システムの構造を記述します。図1に簡単なDFDの例を示します。

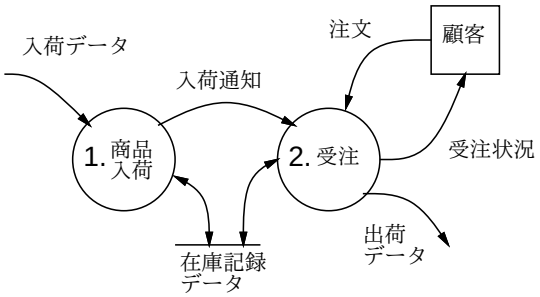

図 1: 簡単なDFDの例

これは何らかの商品の在庫を管理し、顧客からの注文に応えて出荷を行うようなシステムを表しています。ここで「商品入荷」プロセスから「受注」プロセスに「入荷通知」データの流れがある理由が分かるでしょうか。

これはつまり、顧客から注文を受けた時に在庫がなければ、そのことを在庫記録データに記録しておき、商品入荷時に顧客からの受注残のある商品が入ったとわかったらそのことを受注プロセスに知らせる必要があることを意味しています。

しかしそうすると、「受注」プロセスは入荷通知があったときにその商品を予約注文している顧客のデータをどこからか持って来なければなりませんが、そのアテがありませんね? ということは、その情報を格納するデータベースを図に追加する必要があります。

このように、構造化分析では (他の分析手法もそうですが) システムを図で表現して検討することで、見落としや不整合や矛盾を発見して、正しいシステムの構造を作り出せるわけです。

構造化分析で「守らなければならない」DFDの要件として、次のものがあります。

- ○はプロセス (何らかの処理) に対応させ、その処理を表す名前をつける。その中でどのような手順でそれを実行するかはDFDでは記述しない。

- 線分には蓄積されているデータの名前をつける。
- 矢線はデータの流れに対応させる。蓄積場所から出入りする矢線には名前が無くてもいいが (蓄積されているデータと同じになるから)、それ以外の矢線には必ず流れるデータを明記する。
- すべてのプロセスは、その処理を行うのに必要なデータが入って来なければならない。また、その処理の成果を外部に渡すためのデータが出て行かなければならない。

どれも当り前のようですが、情報システムはたいていひどく複雑になるので、全部を一辺に頭に載せることなどできません。だから、間違いなく網羅しようとしても必ず見落としが生じます。そこで、これらの視点をそれぞれ分けてチェックすることで、見落としを発見し、整合性のある正しい DFD に近づけるわけです。

演習 1 次のものからどれかを選び、その主要な構成要素 (プロセス、データベース) を記述する DFD を描いてみなさい。

a. (たとえば TWINS のような) 大学の成績管理システム
b. (たとえば首都高の) 道路情報システム
c. (たとえば楽天トラベルのような) ホテル予約システム
d. (たとえば東京メトロの) 券売機の制御システム

1.4 階層化

前節では 1 枚の DFD でシステム全体を記述できるかのような説明をしましたが、もちろん複雑なシステムではそんなことは無理です。大きな紙で無理矢理描いたとしても、ごちゃごちゃで何が何だか分からなくなるのがせいぜいです。

構造化分析ではこの問題に対して、DFD を階層化することで対処しています。まず、システム全体を 1 つのプロセスとして描いたダイアグラムをコンテキストダイアグラム (図 2) と呼びます。これは、システムが外部とどのようなやりとりをするのか、システムに関与する外部者は何であり、システムからは何が出入りするのかを網羅する役割があります。

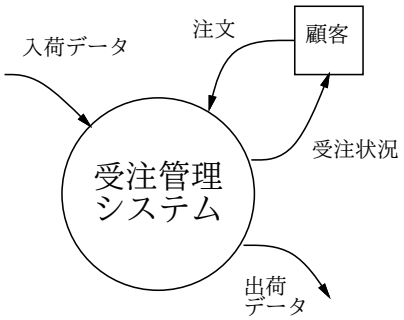

図 2: コンテキストダイアグラム

これを分解した中身が先の図 1 で、これがトップレベルの (レベル 0 の) ダイアグラムとなります。そして、そこに出て来る○ (プロセス) は通常、さらに細かい処理に分解されます。たとえば、1 番の「商品入荷」のプロセスは、さらに 3 つの処理に分解されるかも知れません (図 3)。

こうすることで、上のレベルでは全体的な構成をチェックでき、また下のレベルでは細かい処理の割り振りをチェックできるわけです。そして、必要ならこの各プロセスもさらに細かく分けて行き、具体的な手順として実現できる程度まで分けたら、あとは設計に進んでコードを書けるようにして行きます。

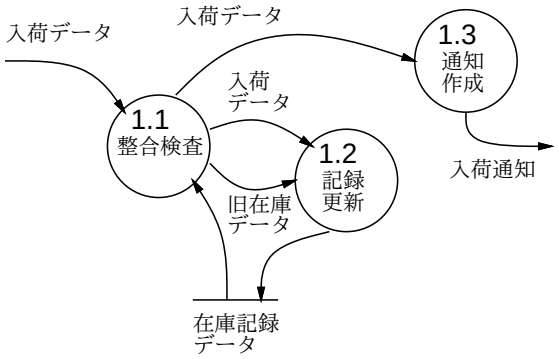

図 3: レベル 1 ダイアグラム

一般にコンピュータサイエンスでは、階層化によって抽象化の高いレベルから低いレベルまでを統一的に扱うという定石がありますが、ここでもそれが適応されているわけです。

なお、このような DFD の階層分割に際しても、守るべき要件があります。

- 下のレベルの処理が全体として、上のレベルの 1 つの○の処理を過不足なく実現していること。
- 上のレベルの○に現れるすべての矢線は、下のレベルにも対応してどこかに現れること。下のレベルに、上のレベルには対応するものが無いような矢線が現れないこと。

各レベルにおいてこのようなチェックを経ていくことで、整合性があり間違いのできるだけ少ない分析をおこなう、というのが構造化分析の要点なわけです。

構造化分析については、デマルコの次の本が定番です (原著は非常に古い本ですが、日本語版は今でもよく売れていて増刷されています)。

> トム・デマルコ著, 高梨・黒田監訳, 構造化分析とシステム仕様, 日経 BP, 1994.
> (原著: Tom Demarco, Structured Analysis and System Specificaion, Prentice-Hall, 1979.)

この本には分析とは何か、構造化分析とは何かからはじまってとても丁寧に分析の進め方が説明されています。たとえば、それぞれの○の中を説明する用語は辞書を作って管理して未定義がないようにするとか、○の中の処理を手続きで実現する (構造化設計) とかまであるのですが、今だとどこかから先はオブジェクト指向でしょうし、そのあたりはちょっと古くなってるかな、と思います。しかし分析自体については今でもまったく古さを感じさせない名著ですので、読んでおくことを薦めます。

演習 2 前問で作成した DFD について、コンテキストダイアグラムを描き、また階層分解してみなさい (どのレベルまでやるかは適当に決めてよいです)。

2 Java によるネットワークプログラミング

2.1 情報システムとネットワーク

今日の情報システムにおいて、ネットワークは不可欠な要素となっています。それには大きく分けて 2 つの理由があります。

1. 情報システムを「情報の発生するその場所で」使えるようにするために (これは事業所などの現場という意味と、顧客が直接自宅などから使うという意味とがあります)、ネットワークが不可欠だから。

2. 情報システムの「内部の構造として」複数の要素がネットワークでつながって全体として動作する、という形のものが一般的になってきたから。

前者についてはとくに説明するまでも無いと思いますが、後者についてはネットワークが広く普及してそのコストが低下し、またコンピュータシステムも 1 台ずつが安くなって、多数のシステムをつなげて動かす方が性能上も構築の易しさからも好ましくなったことによります。

たとえば、先にやった DFD を用いてシステムの構造を記述したとして、それを実際に動くシステムにするときに、これまでは各処理をモジュールとかオブジェクトに変換して組み合わせて動くようにしていたわけですが、そうするかわりに個々の○ (プロセス、処理) をシステム上でもプロセスとして自律実行させ、矢線に対応するネットワーク接続でデータを送受する形でシステムを組み合わせることも可能です (やや極論ですが)。

また、もう 1 つの面として、今日のネット上にはさまざまな有用なサービスが (おもにブラウザから呼び出すために) 提供されていますが、自分のシステムを作るさいにこれらのサービスを利用することで、少ない労力で豊富な機能を持たせることができます。そうなると、自分で作成する部分もいくつかの個々のサービスに分け、それら全てを組み合わせることでシステムとして動かすことが自然です。このような枠組みをサービス指向アーキテクチャ(SOA、service oriented architecture) と呼びます。

以下では、いくつかの例題をもとにこのようなシステムの作り方に少しだけ触れてみたいと思います。

2.2 Java による URL アクセス

とりあえず、今日ネット上で最も多くのリソースにアクセスできるのは Web ですから、Web サイトから情報を取得する簡単なプログラムから始めてみましょう。次のプログラムは、URL を指定するとその URL の中身を取り出して出力するものです。

```
import java.net.*;
import java.util.*;

public class Sample11 {
  public static void main(String[] args) throws Exception {
    System.setProperty("http.proxyHost", "utogw"); // GSSM only
    System.setProperty("http.proxyPort", "8080");  // GSSM only
    System.setProperty("http.nonProxyHost", "w3in|10.*.*.*"); // GSSM only
    URL url1 = new URL(args[0]);
    Scanner sc = new Scanner(url1.openStream());
    while(sc.hasNextLine()) {
      System.out.println(sc.nextLine());
    }
  }
}
```

main() の冒頭の 3 行は GSSM 内から外側のサイトを参照できるようにするためのプロキシ設定と、プロキシ使用から除外するサイトの指定です。たとえば自宅など直接インターネットにつながっている場所で使用する場合はこの 3 行をコメントアウトしてください。

その次からが本体で、まずコマンド引数として渡したも文字列をもとに URL オブジェクトを生成し、そのメソッド openStream() によって生成した読み込みストリームを 1 行単位で読めるように Scanner に持たせます。あとはストリームの終りまで 1 行ずつ読んで出力するだけです。

動かし方は次の通り (Java のプログラムはクラス名と同じ名前のファイル「Sample11.java」に入れる必要があることに注意)。

```
% javac Sample11.java            ←コンパイル
% java Sample11 http://w3in/     ← URL を指定して実行
<HTML>                    ← いきなり HTML が出力される
...
%
```

演習 3 このプログラムをそのまま打ち込んで、動かしなさい。実際にさまざまなサイトの URL を指定して、どんな感じのものが取れて来るか様子をつかみなさい。納得したら、次のような改造をしてみなさい。

a. 取り寄せた HTML 中の「href="...."」という部分だけを抜き出して表示するようにしてみなさい。(ヒント: String のメソッド indexOf()、substring() などを活用する。)
b. 取り寄せた HTML 中に埋められたリンク先を全部取り寄せるようにしてみなさい。(ヒント: 絶対 URL でない URL を解釈するためには、元の URL を指定できる形の URL のコンストラクタを使用するとよい。)
c. GSSM のサイト内のすべてのページの URL 一覧を作ってみなさい。
d. 興味ある外部 URL からつながっているページを一定数 (たとえば最大 100 ページぶん) 収集するようにしてみなさい。
e. 次の URL により、指定した ISBN の書籍に関する情報が取得できる (日本書籍出版協会のサービス)。

   http://www.books.or.jp/ResultList.aspx?searctype=1&isbn=番号&showcount=1&startindex=1

   これを利用して、ISBN(13 桁) を指定したらその書籍の名前を表示するプログラムを作ってみよ。
f. その他、適当な外部サイトのサービスを便利に活用するプログラムを作ってみなさい。

注意: 外部サイトにプログラムアクセスするときは相手に迷惑を掛けないように注意すること。岡崎図書館事件にならないように。

質問 3 このような、Web サービスを下請けとして使用するシステムの利点と弱点について、思い付くものをあげなさい。

2.3 Java によるネットワークサーバ

サービスを利用する側に続いて、サービスを提供する側についてもちょっとだけやってみましょう。最低限の機能だけでブラウザから情報を読める Web サーバのおもちゃを示します。

```
import java.net.*;
import java.io.*;
import java.util.*;

public class Sample12 {
  public static void main(String[] args) throws Exception {
    ServerSocket ss = new ServerSocket(4192);
    System.out.println("starting...");
    int count = 1;
    try {
```

```
      while(true) {
        Socket cs = ss.accept();
        PrintWriter out = new PrintWriter(cs.getOutputStream(), true);
        Scanner sc = new Scanner(cs.getInputStream());
        while(true) {
          String line = sc.nextLine();
          System.out.println(line);
          if(line.equals("")) { break; }
        }
        out.print("HTTP/1.1 200 OK\r\n");
        out.print("Content-type: text/html\r\n");
        out.print("connection: close\r\n");
        out.print("\r\n");
        out.printf("<body><i>Hello %d</i></body>\r\n", count++);
        cs.close();
      }
    } finally {
      ss.close();
    }
  }
}
```

先の例題よりは少し長いので分けて説明して行きます。

- TCP でアクセスされるサーバを作る場合は ServerSocket オブジェクトを作って使います。ここで引数はポート番号であり、1024 未満の番号は特権がないと作れませんが、大きい番号のは自由に作れます。なんとなく 4192 を使っています。
- 何回アクセスしても番号が同じだとつまらないので、1 回ごとにカウンタが増えるようにするので、そのカウンタが count。
- 何かエラーがあって終了するときも必ずサーバソケットが close() されるように try 文で囲んであります。
- while ループの内側が 1 つのアクセスにつき 1 回実行される部分。
- まずサーバソケットに接続されてくるのを accept() で待ちます。接続されると Socket オブジェクトが返されるので、そこから読むためのストリームとそこに書くためのストリームを生成。
- まず、空行が来るまで読みます。この部分は HTTP 要求ヘッダであり、ブラウザからサーバにいろいろ情報を渡しているのですが、ここではあっさり画面に表示するだけであとは無視しています。
- 次に、こちらから HTTP 応答ヘッダとして最低限の情報である状態行と Content-type:ヘッダを送信し、空行を送信してから本体部分 (ここでは 1 行の HTML) を返しています。

では、これを動かしている様子を見てみましょう (エラーが出ない限り終らないのでサーバを止めるには Control-C で停止させてください)。

```
% javac Sample12.java
% java Sample12
starting...
GET / HTTP/1.0
Host: smr06:4192
```

```
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.22)
Gecko/20090606 SeaMonkey/1.1.17
Accept: text/xml,application/xml,application/xhtml+xml,text/html;
q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: ja,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-2022-JP,utf-8;q=0.7,*;q=0.7
Via: 1.1 utogwgw.gssm.otsuka.tsukuba.ac.jp:8080 (squid/2.6.STABLE17)
X-Forwarded-For: 10.2.2.6
Cache-Control: max-age=0
Connection: keep-alive

...

^C  ← Control-C で停止
%
```

ちなみにブラウザ側からは図 4 のように「http://ホスト名:ポート番号/」という URL でアクセスします。再読み込みするごとにカウントが増えて行くのが分かるはずです。

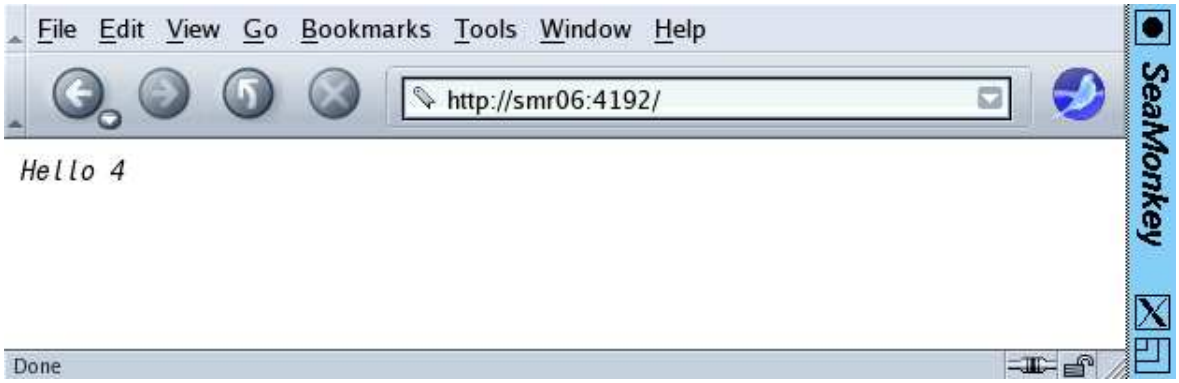

図 4: ブラウザから Sample12 に接続

演習 4 このプログラムをそのまま打ち込んで動かしなさい。動いたら次のような観察を行ってみなさい。

a. 指定する URL をもっと長くしたらどのような違いが観測されるか調べてみなさい。
b. Content-type:として text/html を通知していますが、これを text/plain にするとどうなるか試してみなさい。
c. 今は出力が 1 行だけですが、もっと行数を増やしてみなさい。また、1 行出力するごとに 1 秒、時間をあけるとどうなるかやってみなさい。(ヒント: 1 秒時間待ちをするのには「Thread.sleep(1)」を呼べばよいです。)
d. 返送する HTML に IMG 要素を含めてみて、その結果どういうことが起きるか観察しなさい。

質問 4 ここまでに見て来たことがらから、ブラウザと Web サーバのやりとりについて、どのようなことが分かりますか。また、それは「なぜ」そのようになっているのでしょうか、そして「どのような」利点や欠点があるのでしょうか。

2.4 HTTP プロトコルと Web サーバ

別資料として HTTP 1.0(現在主に使われている 1.1 より 1 つ前のバージョン) の RFC(仕様書) を配布しました。1 つ前のにしたのは、あんまり長いと読む気が失せるからです。

ここまでで (なんとなく) 分かったように、HTTP ではブラウザはサーバに接続し、まずメソッド (GET が代表的) とプロトコルを指定し、続いて HTTP 要求ヘッダによって複数の付加情報を渡します。その後空行があり、もしブラウザからサーバに本体データを渡す場合はその後に本体データが続きます。

サーバ側はこれらの情報を受け取り、まず状態行 (200 OK が代表的を返します。続いて HTTP 応答ヘッダによって複数の付加情報を渡します。その後空行があり、それに続いてブラウザからサーバに渡す本体データが来ます。

これらのプロトコルによって必要な付加情報と本体データを渡し合う、というのが HTTP の全てです (もちろん細かい制御は色々ありますが)。そして、Web サーバは渡された URL に応じて「どこから」「どんな」データを取って来るとか、または必要に応じて内部でどのようなプログラムを動かしてデータを処理させるかなどを全部決めて実装するわけです。本ものの Web サーバはこの部分が非常に複雑で大規模になっていますが、土台となる HTTP の原理は非常に簡単であることがお分かり頂けるかと思います。

演習 5 先の例題サーバを修正して次のような実験をやってみよ。

a. 200 OK 以外のレスポンスにどのようなものがあるか調べ、実際に使ってみよ。(ヒント: リダイレクト系は Location: ヘッダにより転送先を指定すること。)
b. 画像データを生成して送り返す機能を追加してみなさい。渡されたパスに応じてさまざまな画像になるとなおよいです。
c. URL のパス部分をそのままパス名と解釈して、そのファイルを直に送り返すようなサーバを作る。(注意! どのファイルでも取れるというのは実際にはとても危険なので本番運用してはいけません。なぜ危険なのか分からない人は分からせておくこと。)
d. HTML にフォームを入れて、そのフォームのデータを受け取って処理できるような機能を加えてみなさい。
e. その他、ブラウザから使ってみて面白い機能を何か入れてみなさい。

3 まとめ

今回は「情報システムとは何か」という話からはじめて、情報システムを定式化し記述する手段の 1 つである構造化分析と DFD について説明しました。次に、ネットワークが情報システムにとってどのような意味を持つかという話題を取り上げ、最後に Java による実際のネットワークプログラムをいくつか扱い、併せて Web と HTTP などの基本的な仕組みについて見直してみました。これらの基本的な事柄は普段あまり意識しないかも知れませんが、システムの構想・設計やトラブル対処などの際にこれらのことをきちんと把握しておくことは重要ではないでしょうか。