

はじめに

本書はあまり急がないでウェブアプリケーション（ウェブアプリ）の基礎とその周辺知識を学ぼうという人のためのものです。以下に該当する方にお勧めします。

- 基本的なことがわかっていないような気がしている情報系の学生。
- ウェブアプリの作り方を知りたい。
- 実践的な例で Java を勉強したい。
- Eclipse を使いたい。
- MySQL をちゃんと使えるようになりたい。
- 文字コードをちゃんと理解したい。
- ウェブアプリや MySQL の文字化けを解決したい。
- 国際化に対応したソフトウェア開発の基本を知りたい。

ウェブアプリの作成には、総合的な知識が要求されます。いわゆる情報技術 (IT) の初歩として学習するようなプログラミング言語やデータベースを組み合わせるからです。

私たちは、ウェブアプリをそのような総合的な課題として扱っている教科書を探していましたが、ウェブアプリの制作方法を解説した書籍はすでに数多く出版されているにも拘らず、残念なことに、私たちの求めるものはありませんでした。詳しくは 1.4 節 (p.6) で説明しますが、既存の書籍は次のような点で不満でした。

- 扱う範囲が限定されすぎている。
- さらに学びたい人のための情報が少ない。
- 開発環境が貧弱である。
- 動作 OS が限定されている。
- セキュリティや文字コード、HTML の規格に対する配慮が足りない。

膨大な参考文献リストを用意して、そこに挙げた文献を読んでもらうことにすれば、これらの不満は解決できるかもしれません。しかし入門時には、参考文献リストよりも、コンパクトなチュートリアルのほうが学習の効率はいははずです。そこで、このような教科書を作ることにしました。

コンパクトなチュートリアルといっても、本書は、週末に急いでウェブアプリのプロトタイプを作らなければならないような人のためのものではありません。他の入門書に比べると、学ばなければならないことがかなり多いという印象を持たれるかもしれません。しかし、ソフトウェア開発は「1 週間でわかる」とか「10 日でわかる」というものではありません。ですから、あまり急がずに学ぶのがいいと思います。次のような話もあります。

プログラミングを独習するには10年かかる—Peter Norvig（文献 [37]）

私たちがこれに同意します。本書自体は3日もあれば読めるものかもしれませんが、参考文献などをたどって行けば、C言語の基礎しか知らない方が1年ぐらいは楽しめるようにしたつもりです。

使用ソフトウェア

本書で主に使用しているソフトウェアを挙げます¹⁾。

- Mozilla Firefox
- Java SE Development Kit (JDK)
- Apache Tomcat
- MySQL
- Eclipse + Web Tools Platform (WTP)

使用 OS は以下の通りです。

- Windows XP, Vista
- GNU/Linux
- Mac OS X 10.3 以上

本書で利用するソフトウェアをすべてインストールした GNU/Linux のライブ CD 用のイメージファイルを用意しました。開発環境の準備が面倒な場合や、普段使っている PC にソフトウェアをインストールしたくない場合にご利用ください（第2章 p.13 を参照）。

本書についての質問や要望は、著者 (taro.yabuki@unfindable.net) にお送りください。サポートサイト (<http://www.morikita.co.jp/soft/84731/>) で対応します。このサイトでは、本書に掲載されているコードも公開しています。

2007年7月

著者

1) 動作確認したソフトウェアのバージョンは、Firefox 1.5 および 2.0, JDK 5.0, Tomcat 5.5, MySQL 5, Eclipse 3.2, WTP 1.5, OS は Windows XP, Vista, Mac OS X 10.3 および 10.4, GNU/Linux (Slax 5.1.7b-2 および 5.1.8.1-3) です。これらより新しいバージョンについては、できるかぎりサポートサイトで補足します。

目次

第1章	本書の目指すもの	1
1.1	ウェブアプリとは何か,なぜウェブアプリなのか	1
1.2	本書を読む上での注意	3
1.3	本書が前提にしている知識	5
1.4	本書の意義	6
1.5	ガイドマップ	9
第2章	クイック・スタート	13
2.1	ライブ CD の作成	13
2.2	ライブ CD の利用	13
第3章	ウェブページの書き方 1	16
3.1	ウェブブラウザ—Firefox	16
3.2	HTML 入門	17
第4章	ウェブページの書き方 2	20
4.1	主な HTML 要素	20
4.2	XHTML	22
4.3	CSS: ウェブページの外観	26
4.4	ウェブサーバ上の HTML 文書	33
第5章	データの送信	36
5.1	サーバの準備	36
5.2	フォーム	38
5.3	GET: URL によるデータ送信	42
5.4	POST によるデータ送信	43
第6章	ダイナミックなページ生成	45
6.1	JSP: HTML と Java の融合	46
6.2	リクエスト内容の取得	52

6.3 スクリプトレットを使わない JSP	53
第7章 データベースの操作 1	57
7.1 DBMS の必要性	57
7.2 リレーショナル・データベース管理システム	58
7.3 データベースとテーブルの作成	59
7.4 データの操作	63
7.5 アクセス権限	68
7.6 phpMyAdmin	69
第8章 データベースの操作 2	71
8.1 SELECT 文の詳細	71
8.2 文字コードの設定	73
8.3 インポート・エクスポート	77
8.4 テーブルの結合	78
8.5 MySQL でサポートされる関数	91
8.6 グラフの操作	93
8.7 インデックス	99
8.8 ストアド・プロシジャ	101
8.9 カラムに付加する性質	104
第9章 データベースへの接続	109
9.1 JDBC: Java とデータベースの架け橋	109
9.2 JDBC の利用	110
9.3 スクリプトレットでの実装	113
9.4 SQL インジェクション対策	114
第10章 ウェブアプリの例	119
10.1 郵便番号検索	119
10.2 GET による検索	121
10.3 フォームからの検索	122
10.4 Ajax によるインターフェースの改良	122
10.5 ウェブアプリの配備	123
第11章 Model, View, Controller	128
11.1 MVC とは何か, どう実装するのか	128
11.2 郵便番号検索の MVC による実装	129
11.3 プロフィール登録システム	134

第 12 章 国際化	142
12.1 国際化とは	142
12.2 ウェブアプリの国際化	143
付録 A C プログラマのための Java	147
A.1 オブジェクト	147
A.2 例外	151
A.3 クラスライブラリ	153
付録 B 文字コード	167
B.1 文字コードとは	167
B.2 どの文字集合を使うべきか	169
B.3 文字コードの統一	171
B.4 ウェブブラウザが利用する文字コード	174
B.5 アプリケーションサーバでの処理	176
付録 C 開発環境の構築	178
C.1 Windows Vista および XP	178
C.2 GNU/Linux	197
C.3 Mac OS X	207
参考文献	212
索引	216

CAPTER 1

本書の目指すもの

本書はウェブアプリケーションの作り方を学ぶためのものです。そもそも、ウェブアプリケーションとは何でしょうか。その作り方は、どのように学んだらいいのでしょうか。類書が多くある中で、本書の意義はどこにあるのでしょうか。本章では、これらの疑問に答えます。

1.1 ウェブアプリとは何か、なぜウェブアプリなのか

ウェブアプリケーション（ウェブアプリ）とは何でしょうか。通常のウェブページとの比較で説明しましょう。

ウェブページは、ネット上で情報を公開する方法の一つです。

図 1.1 を見てください。情報を公開したい人（制作者）は、その情報を書いたウェブページを作成し、ウェブサーバ上に置きます。クライアント（ウェブページを見たい人）は、そのウェブページのアドレスをウェブサーバに通知し、閲覧を要求（リクエスト）します。ウェブサーバは要求のあったウェブページをクライアントに送信します。

ここで重要なことは、この形式では制作者があらかじめ作っておいたウェブページしか、クライアントに提示できないということです。そのため、クライアントとサーバのインタラクションはとても単純です。

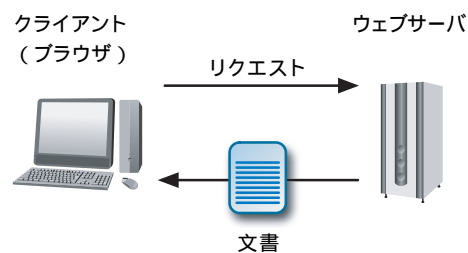


図 1.1 ウェブページの閲覧

ウェブアプリもネット上で情報を公開する方法の一つですが、ウェブページと異なり、情報公開にとどまらず、クライアントとサーバの複雑なインタラクションを可能にします。クライアントからの要求に対して返せるウェブページは、あらかじめ作っておいたものに限られません。要求を受けてからウェブページを生成し、それをクライアントに返すことができます。

これを可能にするのが図 1.2 のようなシステムです。ウェブサーバはクライアントからの要求をアプリケーションサーバに転送し、アプリケーションサーバがウェブページを動

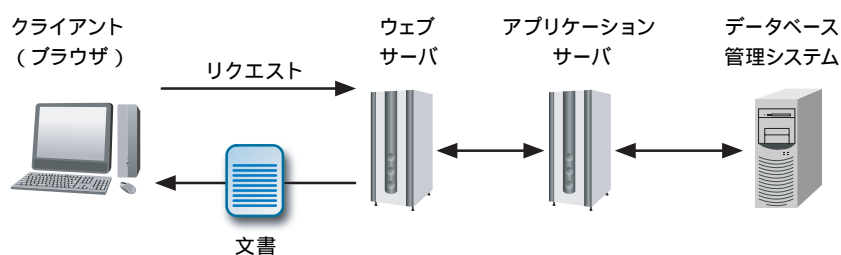


図 1.2 ウェブアプリの一般的な構成

的に生成します。処理に必要なデータは、データベースに格納されるのが一般的で、データベース管理システム (Database Management System, **DBMS**) がその管理を請け負います。

「アプリケーション」ということに関して、メールクライアント (メーラー) を例にもう少し説明しましょう。

かつてメーラーと言えば、クライアント・コンピュータにインストールするスタンドアロン・プログラムしかありませんでした。しかし、今ではウェブメールが広く普及しています (ウェブ上でしか利用できないメールサービスさえあります)。ウェブメールにはさまざまな利点がありますが、最も重要なのは次の 2 点でしょう。

1. インターネットに接続したコンピュータとウェブブラウザがあれば利用できます。ソフトウェアをインストールする必要はなく、ブラウザが動くなら OS も問いません。
2. プログラムをアップデートするのが簡単です。スタンドアロン型のメーラーの場合、新しいバージョンをダウンロード・インストールする必要があります。

最初の点は特に重要です。今日では携帯電話にも標準的なブラウザが搭載されていますから、アプリケーションをウェブアプリの形で作成すれば、パソコンからでも携帯電話からでも利用可能になるのです¹⁾。

ウェブメールには欠点もあります。

1. ネットワークの速度がボトルネックになって、動作が全体的に遅くなることがあります²⁾。
2. インターネットに接続しなければ使えません³⁾。
3. ユーザ・インターフェースが貧弱です⁴⁾。

1) 「一度コードを書けばどこでも動作する (Write Once, Run Anywhere, **WORA**)」というのはソフトウェア開発の一つの理想で、プログラミング言語 Java はこの理想を掲げて登場しました。皮肉なことに、アプリケーションの動作環境として携帯電話まで考えると、Java はこの理想を達成できているとは言えません。しかし、Java などの技術に支えられたウェブアプリなら、この理想を達成できるのです。

2) クライアント・コンピュータ内での通信速度とインターネットの通信速度の差を考えれば、動作が遅いということに関しては、あきらめるしかありません。とはいえ、ネット上にあることによって、ウェブアプリは個人の PC には格納できないような膨大な情報を利用することができます。

3) 最近では、コンピュータはインターネットに常時接続しているのが当たり前になっていますから、これは欠点ではなくなりつつあります。また、ブラウザをユーザ・インターフェースにするだけなら問題ありません。

4) **JavaScript** や **Flash** によってユーザ・インターフェースを改善する試みが続けられているため、将来この欠点は解消されるかもしれません。

このような欠点があるとはいえ、インターネット上での利用を想定しないようなアプリケーションでさえも、ウェブアプリとして実装する価値はあります。新たに学ばなければならないことはありませんし、後でインターネット上で利用しようということになったときに、アプリケーションを修正する必要もありません。たとえば、7.6節 p.69で紹介する phpMyAdmin は、インターネット上でデータベース管理システムを操作できるアプリケーションですが、そのような遠隔操作を必要としない人にとっても、使いやすいものになっています。

まとめると、「ウェブアプリは、サーバとクライアントの間の複雑なインタラクションを可能にする技術で、その枠組みはウェブに限らず一般的なアプリケーションに用いることもできる」ということとなります。

1.1.1 なぜウェブなのか

少し話を広げて、なぜウェブ (World Wide Web, WWW, Web) なのかということを考えてみましょう。

インターネットの普及の速度は驚異的です。地上に限らず、人類の到達できるほとんどの場所でインターネットを利用できるようになっています。ウェブは、狭義にはインターネットの一つの利用形態あるいはプロトコルですが、広義にはインターネットとほとんど同じ意味になっているでしょう。

ウェブ上には膨大な知識が蓄積され、日々新しいサービスが展開されています。そのウェブと人間の間にあるのがウェブアプリです⁵⁾。どんなに知識が蓄積されても、それが人間にとってアクセスしやすいものになっていなければ何の価値もありません。ですから、ウェブのユーザ・インターフェースであるウェブアプリをよりよくすることは、とても重要なことです (この議論は単純化しすぎています。知識とインターフェースは分離できません⁶⁾)。

このように、ウェブは人類の営みの重要な要素となっています。その重要な構成要素であるウェブアプリについて学んでもらうのが本書の目的です。

1.2 本書を読む上での注意

本書はウェブアプリについての入門書ですが、入門的ではない事柄も含まれています。それらは、各コラムや章末の「より詳しく知りたいときは」で扱っています。初めのうちは、これらはすべて無視してかまいません⁷⁾。文献を参照する [1] のような表記も無視してください⁸⁾。全体像を把握するのが先決です。

5) ウェブアプリと明確に区別できるわけではありませんが、人間ではなくプログラムに対するインターフェースがウェブサービスです。ウェブサービスは本書では扱いません。

6) このような視点からウェブについて深く考えたい場合は、文献 [33] が参考になります。同じ著者の文献 [38] もウェブ上の情報のあり方を考える際の必読書です。

7) 脚注も初めは無視してかまいません。

全体像の把握のためには、図 1.3 (p.10) のガイドマップも活用してください。読み進めていくうちに、自分が何をしているのかわからなくなったときは、ガイドマップに戻ってみてください。

バックスラッシュの字体は左上から右下への斜線を使っています（使用書体によって多少変わりますが、字形は「\」のような形です）。日本の通貨記号（**円記号**「¥」）と同じ字形が使われることがあります。バックスラッシュと円記号を区別できなくなるため本書では使いません⁹⁾。

付録 B.1 (p.167) で詳しく説明しますが、**Windows-31J** というのは**文字コード**のことです。**Code Page 932 (CP932)** や **MS932** と呼ばれることもあります。この文字コードは、間違っ**て Shift_JIS** と呼ばれることが多いのですが、Windows-31J と Shift_JIS は別のもので**す**¹⁰⁾。

Windows や Mac OS X と違って、GNU/Linux にはさまざまな形態があります。本書で想定している GNU/Linux は 2 章 (p.13) で紹介しているものです。自分の GNU/Linux システムに開発環境を構築した場合には、設定方法や動作が本書とは異なる可能性があります。深刻な違いはないはず**です**。

本書の中でウェブ上の資料を紹介する際には、その URL と併せて掲載しています。URL が変わったり資料がなくなったりすると、URL を使ってその資料を閲覧することはできなくなります。そのような場合には、資料のタイトルで検索すれば、新しい URL や検索サイト上のキャッシュが見つかるでしょう¹¹⁾。

表 1.1 本書の読み方の例（△：コラムと脚注をとばす）

	コース1	コース2	コース3
1 章 本書の目指すもの	△	△	○
2 章 クイック・スタート	付録 C でもよい	付録 C でもよい	○
付録 C 開発環境の構築	2 章でもよい	2 章でもよい	○
3 章 ウェブページの書き方 1	△	△	○
4 章 ウェブページの書き方 2		△	○
5 章 データの送信	△	△	○
付録 A C プログラムのための Java	△	△	○
6 章 ダイナミックなページ生成	△	△	○
7 章 データベースの操作 1	△	△	○
8 章 データベースの操作 2		△	○
9 章 データベースへの接続	前半のみ	△	○
10 章 ウェブアプリの例	△	△	○
11 章 Model, View, Controller		△	○
12 章 国際化		△	○
付録 B 文字コード		△	○

8) 参考文献は p.212 以降にまとめてあります。

9) キャプチャ画像には、バックスラッシュの字形が円記号になっているものがあります。

10) 符号化方式という意味ではどちらも Shift_JIS です。

11) Internet Archive (<http://www.archive.org/index.php>) のようなサービスで見つかる資料もあるでしょう。

1.2.1 本書の読み方

先に述べた注意をふまえて、本書の読み方を 3 種類提案します (表 1.1)。これらのうちのどれかを選択するというのではなく、まずはコース 1 で概要をつかんで、それから初めに戻ってコース 2 で読んでいくという読み方でもいいでしょう。もちろん、これらにとらわれず好きなように読んでもらってかまいません。

1.3 本書が前提にしている知識

本書の読者には次の前提知識があることを想定しています。

- ネットサーフィンができる (ネットで検索する方法やハイパーリンクとは何かなどということは説明しません)。
- ファイルやディレクトリの基本的な操作ができる (ネットからファイルをダウンロードしたり、圧縮されたファイルを展開することも含みます)。
- C 言語の基本的な文法を知っている。

1.3.1 前提とする C 言語の知識

C 言語の文法については少し詳しく説明しましょう。

本書で前提としているのは、C 言語 (あるいは C++ や Java) の基本的な知識です。C 言語の初学者がまずくポインタや文字列についての詳しい知識は必要ありません (知っていればより良いプログラムが書けるかもしれませんが)。たとえば、文献 [5] の第 1 章「やさしい入門」がわかっているだけで十分です。

いくつか例を挙げましょう。これが何かわかりますか？

```
int s=0;
```

整数型の変数 s を宣言し、値 0 で初期化しています。

次はどうでしょうか。

```
int a[]={1,3,5};
```

$a[2]$ の値は 5 です (配列の添え字は 0 から数えます)。

For 文はどうでしょう。

```
for(int i=1; i<=100; ++i) s+=i;
```

変数 s の値は 1 以上 100 以下の整数の和になります (本書では 1 行しかないブロックでは {} を省略します)。

If-Else 文はどうでしょう。

```
if(s%2==0) printf("even\n");  
else printf("odd\n");
```

変数 s を 2 で割った余りが 0 (つまり 2 で割り切れる) なら “even” と、そうでないなら “odd” と表示します。

最後は関数です。

```
int sum(int start, int end){
    int s=0;
    for(int i=start; i<=end; i++) s+=i;
    return s;
}
```

これは、整数 $start$ から end までの和を返す関数です。使い方はわかりますか。

説明を読まないと意味がわからないものがあるなら、本書を読む前に C 言語の復習をしてください (C 言語を学んだことがないなら、C 言語ではなく Java の入門書を読んでもいいでしょう。ただし、薄い本を選ぶこと)。

1.4 本書の意義

「はじめに」で述べたように、私たちは、情報系の総合的な演習としてウェブアプリの制作方法を学ぶための教科書を探していました。候補となる書籍はたくさんありましたが、次のような点において不満を感じるものでした。

- 扱う範囲が限定されすぎている。
- さらに学びたい人のための情報が少ない。
- 開発環境が貧弱である。
- 動作 OS が限定されている。
- セキュリティや文字コード、HTML の規格に対する配慮が足りない。

詳しく説明しましょう。

プログラミングの入門課程を修了している人が、ウェブアプリを制作しようというときに必ず学ばなければならないのは次のようなことで、多くの書籍は、これらをコンパクトに紹介するだけにとどまっています (つまり、扱う範囲が限定されすぎています)。

- ユーザと対話する画面を記述するための HTML と CSS
- サーバ側での処理を記述するためのプログラミング言語
- データを蓄積するためのデータベース管理システム¹²⁾
- これらの要素をつなぎ合わせる方法

掲載されているサンプルをそのまま入力して、本に載っている手順通りにセットアップすればウェブアプリが動く、というのが目的であれば、その達成は難しいものではありません。しかしながら、いざ自力でウェブアプリを作ろうという段階になると、次のような問題が発生します。

12) 多くのウェブアプリの入門書では、データベース管理システムは単にデータを保管する役割しか担っていません。しかしながら、データベース管理システムでできることはそれにとどまりません。使いこなせるようになれば、ウェブアプリ以外の場面でも、強力なツールになるでしょう。

1. プログラミング言語の知識が足りないために挫折する。
2. 開発環境が貧弱なために、開発が困難になる。
3. ウェブアプリやデータベースの設計方法を知らないために非常に複雑な実装になる。
4. セキュリティ・ホールだらけのウェブアプリを作成してしまう。
5. 文字化けを解決できない。

すべてを網羅した完璧な入門書というのはありませんから、問題が発生するのは仕方ありません。しかし、入門書では解決できないような問題が発生したときに、何か勘が働くようにはなっていたいものです。そのためには、入門書の中にたくさんの「さらに学びたい人のための情報」が必要です。

1.4.1 プログラミング言語の知識

1 番目の問題「プログラミング言語の知識が足りないために挫折する」は、「C 言語は知っているけど Java はよく知らない」という人によく起こります。ウェブアプリを制作しようとして Java の知識が必要であることを知り、Java を勉強しようと分厚い教科書を買ってきて、その分厚さに挫折する、というわけです。プログラミングの教育が、C 言語ではなく Java のようなオブジェクト指向言語で始まるようになれば、このような問題は起こらないのかもしれませんが、そうはなってはいないのが現実です。

ここに一つの提案があります。ウェブアプリ制作を Java の入門教材として利用するので、C 言語を知っている人が、Java を勉強するために分厚い教科書を通読する必要はありません。Java (あるいは Java 型の**オブジェクト指向プログラミング言語**)のエッセンスだけを新たに学ばばよく、そのための題材としてウェブアプリは最適なものの一つだと思われれます。

オブジェクト指向プログラミングの勉強ということを別にすれば、ウェブアプリの基礎を学ぶ際に、プログラミング言語についてこだわる必要はありません。ウェブアプリは一つのプログラミング言語を知っていれば作れるというようなものではありません¹³⁾、どんなプログラミング言語も、今日の形のままで使われ続けるということはないはずだからです。ですから、「データ構造とアルゴリズム」や「正規表現」のような、プログラミング言語が変わっても廃れないような知識を身につけることが大切です。

本書では、以上のことを考慮した資料として、「C プログラマのための Java」という章を設けました(付録 A p.147)。

13) 今日のウェブアプリでは、サーバ側で動く Java (あるいは PHP, Perl, Ruby など) と SQL, クライアント側で動く JavaScript と、少なくとも 3 つのプログラミング言語が使われます。Google Web Toolkit (<http://code.google.com/webtoolkit/>) のようなクライアント側も Java で書けるようにする試みや、Java 6.0 で導入されたスクリプト・エンジン (**javax.script**) によってサーバ側で JavaScript を使えるようにする試みもありますが、SQL を Java や JavaScript のような汎用言語で置き換えることはできないでしょう。

1.4.2 貧弱な開発環境

2 番目の問題「開発環境が貧弱なために、開発が困難になる」は、適切な開発環境を与えられていないために起こります。一部の入門書は、テキストエディタとコマンドプロンプトを開発環境として利用しています。これは、今日ふつうに利用できるウェブアプリ開発環境と比較して、あまりに貧弱です。早いうちから電卓を使うと計算能力が身につかないというのと同じように、便利な開発環境のために、身につかない知識というものもあるでしょう。しかしながら、ウェブアプリ開発の最先端ははるか遠くにあるので、入門書の段階でなくてもいい苦勞をすべきではありません。そもそも Java 型のオブジェクト指向プログラミング言語は、シンプルなテキストエディタで開発できるものではありません。

本書では、Java を用いる開発における**統合開発環境** (Integrated Development Environment, **IDE**) のデファクト・スタンダードになりつつある **Eclipse** と¹⁴⁾、そのウェブアプリ開発用プラグインである **Web Tools Platform (WTP)** を用いることにしました。

1.4.3 複雑な実装

3 番目の問題「ウェブアプリやデータベースの設計方法を知らないために非常に複雑な実装になる」が起こるのは、入門書なのだから仕方ないという見方もあるでしょう。ウェブアプリやデータベースには、ある決まった設計方法（ウェブアプリには **MVC**、データベースには **正規化** など）があり、その枠組みに合うようにすれば、システムの構築や保守、拡張が容易になるとされています。とはいえ、これらの設計方法は、入門書の最初のサンプルになるほどには簡単ではないため、あまり扱われません。入門書をもとに独自のウェブアプリを作成する場合、最初に覚えた簡単なサンプルを拡張しようとすることになるのですが、そういう方法では、システムが少し大きくなると、手に負えないくらい複雑なものになってしまいます。入門書の性格上、手軽なサンプル以上のものは扱えないということはあるでしょう。しかしそうならば、「さらに学びたい人のための情報」として、このような話題を提供したいものです。

本書では、**MVC** については多くの入門書よりも詳しく説明しました。それでも、本書で扱いきれない事柄はたくさんあります。それらのために、各章末に「より詳しく知りたいときは」という節を設けました。

1.4.4 セキュリティ・ホール

4 番目の問題「**セキュリティ・ホール**だらけのウェブアプリを作成してしまう」は、入門書だから仕方ないと済ますわけにはいきません。安全なウェブアプリを構築することは、簡単なウェブアプリを構築することに比べて、はるかに難しいことです。まして、「絶対

14) Eclipse のほかに広く普及している IDE としては、Sun Microsystems が開発をサポートしている **NetBeans** (http://www.netbeans.org/index_ja.html) があります (文献 [56] を参照)。定評のある IDE なので試してみると良いでしょう。差異を知ることによって本質の理解が深まるということもあります。これはプログラミング言語にも言えることです。

安全」などと言うのは不可能です。それでも、そのままコピーして使うと重大なセキュリティ・ホールになることが明らかなサンプル・プログラムを載せておいて、改善策に触れないというのは問題です。

本書で扱う例においては、ウェブアプリのよくある脆弱性である **SQL インジェクション**と**クロスサイト・スクリプティング**への対策はできるようになっています。セキュリティ対策のための参考文献も紹介しています。

1.4.5 文字化け

5 番目の問題「**文字化け**を解決できない」は、多くの入門書が文字コードを常に意識するようには書かれていないために起こります。ウェブブラウザ上で起こるウェブページの文字化けは、ブラウザの設定を変えるだけで解決できます。しかし一般には、文字化けが起こるとデータが失われ、失われたデータは戻ってきません。つまり、文字化けは想像されるよりも重大な問題なのです。それにも拘わらず、多くの入門書では文字コードの設定を簡単に済ませるか省略してしまっています。ウェブアプリには文字コードを意識しなければならない要素がたくさんあるため、それらを完全に把握しておかないと、「なんだかよくわからないけど動いた」という段階を超えることができません。

本書では、文字コードを設定しなければならないところでは、必ず立ち止まってそれを確認するようにしています。この過程を通じて文字コードについての理解を深めれば、ウェブアプリ以外のところで文字化けに出会っても、適切に対応できるようになるはずで、利用する文字コードを、Windows 環境でよく使われる **Windows-31J** に固定してしまっている書籍も多く見られますが、特に国際化という視点から見たとき、文字コードは **UTF-8** に統一しなければなりません。

1.5 ガイドマップ

ウェブアプリの基本として学ぶことを図 1.3 にまとめました¹⁵⁾。

ウェブアプリは図 1.3 に示したような要素を組み合わせることで実現します。この図の要素に対応する章では、そのことを最初に強調するので、道に迷わないようにしてください。何をしているのかわからなくなったときは、このマップで確認してください。

各要素について簡単に説明しましょう。

HTML 文書：ユーザに何をどのように見せるかはここで決まります（第 3, 4 章）。

リクエスト：ユーザの操作・要求・データをサーバに送信します（第 5 章）。

ウェブアプリケーションサーバ：ユーザからのデータを受け取ることから結果（ページ）をクライアントに返すことまで、ウェブアプリの中核を担います（第 6 章）。

15) ウェブサーバとアプリケーションサーバは別にするのが一般的です。よくあるのは、ウェブサーバが **Apache HTTP Server**、アプリケーションサーバが **Apache Tomcat** という構成です。本書では両方を Tomcat が兼ねています。これが図 1.3 と図 1.2 (p. 2) の違いです。ちなみに、“Apache” という語は、Apache Software Foundation およびそのソフトウェアブランド、Apache HTTP Server の 3 つの意味で使われます。

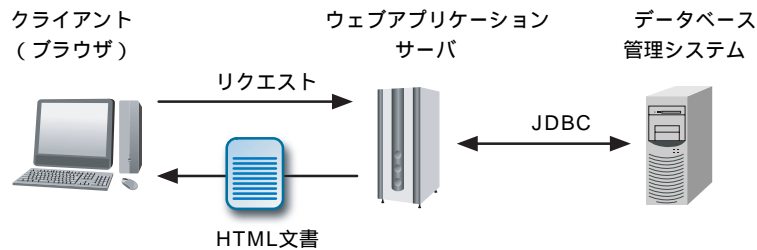


図 1.3 ガイドマップ

データベース管理システム：ウェブアプリに必要なデータを格納します（第 7, 8 章）。

JDBC：ウェブアプリケーションサーバとデータベース管理システムを仲介します（第 9 章）。

ある程度学習が進んだと思ったら、次のようなチェックリストで自問してみるとよいでしょう。

1. HTML 文書（CSS を含む）の書き方がわかるか？
2. ブラウザからサーバにデータを送信する方法がわかるか？
3. 送信されたデータをサーバで受け取る方法がわかるか？ 日本語を受け取れるか？
4. データベース管理システムを使えるか？
5. アプリケーションからデータベースを利用できるか？

これらが基本だというのは、本書のような Java によるウェブアプリに限りません¹⁶⁾。ですから、Java 以外のプラットフォームを使う場合でも、全てを勉強し直す必要はありません。たとえば PHP（コラム参照）でしたら、項目 1, 2, 4 は Java と共通です。項目 3, 5 についてのみ PHP での書き方を確認すれば、簡単なウェブアプリなら実装できるでしょう。

COLUMN 参考

ウェブアプリのプラットフォーム

ウェブアプリのプラットフォームには次のようなものがあります。

- **.NET**
- **LAMP** (Linux+Apache+MySQL+PHP, Perl or Ruby)
- **J2EE** (Java 2 Enterprise Edition)

厳密に言えば、これらは同列に議論するものではありません。たとえば、.NET はアプリケーション・フレームワークとしての ASP.NET を含んでおり、これと比較するなら、J2EE には Struts や JSF、Ruby には Rails などといったアプリケーション・フレームワークを追加すべきでしょう。

他にも多くのプラットフォームがありますが、はじめてウェブアプリを構築する人は、この 3 つのうちからプラットフォームを選択するといいいでしょう。理由は 2 つあります。第 1 に、

16) Ruby on Rails のような高級な環境では、図 1.3 に示したような理解がなくてもウェブアプリを作り始めることはできますが、環境の内部がどうなっているかについて、ある程度直感が働くようになっていないと、使いこなすのは難しいでしょう。

世界中で利用された実績があり、「何かを作ろうとしたが、本質的に無理だった」ということにはおそくなりません。第2に、広く使われているため、周りに必ず詳しい人がいます。

では、この3つのプラットフォームからどれを選んだらよいのでしょうか。どのプラットフォームにも長所・短所があります。そのため、何が最善かを一概に言うことはできません。宗教論争になるだけです。

とはいえ、本書ではJ2EE (Java) を利用します。理由はいくつかあります。

- OS を限定しません。 .NET はこの点でやや難があります。 GNU/Linux や Mac OS X 上で .NET アプリケーションを開発・運用するためのオープンソース・ソフトウェア **Mono** (<http://www.mono-project.com/>) がありますが、入門書で扱うことには抵抗があります。ただし、.NET が私企業のもので、Java はそうではないというのはまったくの誤解です。 .NET の中核言語である **C#** は標準化されています (ECMA-334 や JIS X 3015)。
- オブジェクト指向プログラミング言語の入門を兼ねられます。 **Perl** や **PHP** の場合、オブジェクト指向ということ意識しなくてもウェブアプリを作れるので、この点でやや難があります。 ちなみに Perl や PHP はパフォーマンスが悪いという話がかつてはありましたが、今日ではそういうことはなさそうです ([75] を参照)。
- 開発環境のほとんどすべてを **オープンソース・ソフトウェア** (ソースコードが公開されており、そのコードを改変・再配布することができるソフトウェア) で揃えられます。 オープンソース・ソフトウェアには、無料で導入できるということだけでなく、深く知りたくなったときはソースコードを参照でき、さらには改変することもできるという利点があります。
- 国際化や文字コードについての資料が充実しています。
- 産業界での需要があります (本書は大学のカリキュラムで利用されることも想定しているので、これは無視できません)。

開発現場に行けば、利用するプラットフォームはすでに決められている場合が多いでしょうから、普段から自分でいろいろ試しているのはいいことです (何を試すかは、各章の「より詳しく知りたいときは」を参考にしてください)。ただし、HelloWorld 程度のことしか試さないのでは、あまり意味がありません。複数のプラットフォームを試すならば、最低限 **MVC** (第11章 p. 128) の実現方法がわかる段階までは学んでおきましょう。

COLUMN 参考

本書の内容に関連する資格

対象範囲が本書で扱う内容に近い資格には次のようなものがあります。実際に資格試験を受けないまでも、書店や図書館で教科書や参考書などを見れば、どの程度の知識や技術が資格として認定されているかがわかるでしょう。

Sun 認定 Web コンポーネントディベロッパ: サーブレットや JSP を扱う能力を認定する資格です (<http://suned.sun.co.jp/JPN/certification/compdetails.html>)。

情報処理技術者: 情報処理技術者試験 (<http://www.jitec.jp/>) の個々の試験が扱う範囲はとも広く、その大部分が本書の範囲と重なるようなものはありません。しかしながら、基本情報処理技術者とテクニカルエンジニア (情報セキュリティとシステム管理)・ソフト

ウェア開発技術者・テクニカルエンジニア（データベース）で問われるデータベースに関する知識には本書の内容と重なるものがあります。特に、基本情報処理技術者のものに関しては本書でほぼカバーできるでしょう。それ以外の資格においては本書で扱わない事柄も問われますが、本書を通じて得られる体験は、それらを学ぶ際に役立つでしょう。

ORACLE MASTER : Oracle Master Silver Fellow (Oracle 9i) や ORACLE MASTER Bronze (Oracle 10g) では SQL とデータベース管理者 (Database Administrator, DBA) について問われます。このうち、SQL に関しては本書でほぼカバーしています。DBA は Oracle に特化しているので本書の範囲外です (<http://www.oracle.co.jp/education/master/>)。

MySQL Certifications : MySQL を扱う能力を認定する資格です (<http://www.mysql.com/certification/>)。



クイック・スタート

「はじめに」で述べたように、ウェブアプリの開発環境を準備するのは少し面倒なので、ライブ CD (GNU/Linux) 用のイメージファイルを用意しました。このライブ CD からコンピュータを起動すれば、準備作業をすべてとばして、すぐに本題に入ることができます¹⁾。ただし、このライブ CD を Mac 上で直接起動することはできません。

2.1 ライブ CD の作成

ライブ CD を作成します。

1. サポートサイト (<http://www.morikita.co.jp/soft/84731/>) からライブ CD の ISO イメージ・ファイルをダウンロードする。
2. ISO イメージ・ファイルをもとに CD を作成する (ファイルを、単に CD-R に書き込むのではなく、ディスクイメージとして使う。作成した CD を閲覧すると、LICENCE や COPYING というファイルがあるはず²⁾)。

2.2 ライブ CD の利用

作成したライブ CD でコンピュータを起動すると、図 2.1 のような画面になります。

ユーザ名 root、パスワード toor でログインして、startx とするとウィンドウシステムが起動します。このシステムは、本書で利用する各種ソフトウェアの設定がすでに済んだものです。ターミナル (後述) は「メニュー ⇒ システム」に、Eclipse と Firefox は「メニュー ⇒ インターネット」にあります (デスクトップのパネルにも登録してあります)。

ライブ CD は CD-ROM ですから、そのままでは作成・変更したファイルなどを保存することができません。変更は USB メモリやハードディスクに保存します。

ここでは、USB メモリに変更内容を保存してみます。ターミナル上で「touch test」として新たにファイルを作成し、この変更を USB メモリに保存します。システムを再起動して、この変更 (つまり作成したファイル) を復元できることを確かめましょう。

1. USB メモリをコンピュータに接続する。
2. USB メモリが認識されると新たにダイアログが出現する (とりあえず、「新しいウィンドウで開く」でいいでしょう)。

1) 開発環境の構築自体も勉強になりますから、付録 C (p.178) を参考に試してみるのもいいと思います。

2) USB メモリから起動したり仮想マシン上で利用する場合には、CD を作成する必要はありません。

CAPTER 3

ウェブページの書き方1

ウェブアプリはユーザーに文字や画像・音声などさまざまなものを提示します。それらをどう提示するかを決めるのが **HTML** と **CSS** です。本章では、これらについて学びます (図 3.1)。

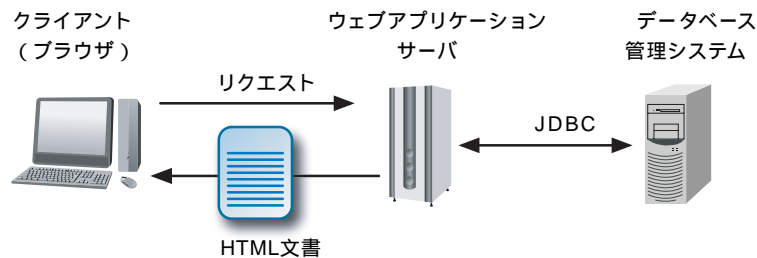


図 3.1 本書で学ぶこと：HTML 文書の書き方

3.1 ウェブブラウザ—Firefox

ウェブアプリのクライアントはウェブブラウザです。本書で想定するウェブブラウザはいわゆるモダン・ブラウザ (Windows 版 Internet Explorer と Opera , Firefox , Safari の最新版ということにしてもいいでしょう) で、ウェブアプリのクライアントとしては、これ以上こだわることはありません。逆に範囲を狭めて、特定のウェブブラウザでしか動作しないようなウェブアプリを作らないようにしてください。

しかしながら、学習や開発で利用することを考えると Firefox が最適です¹⁾。それは次のような理由からです。

- 無料で自由に使える。
- ウェブアプリ開発に役立つ多くの拡張機能を無料で自由に使える。
- プラットフォームを限定しない。つまり、Windows と GNU/Linux, Mac OS X で動作する。

❖ 付録 C (p.178) を参考に、Firefox をインストールしてから先に進んでください。

1) アプリケーションの動作確認のためには、Opera もインストールしておくといでしょう (フリーソフトウェアではありませんが無料です)。

CAPTER 4

ウェブページの書き方2

本章では、前章に引き続いてウェブページの書き方を説明します。前章に比べて細かい事柄が多くなるので、本書を初めて読む際や、とりあえずウェブアプリの全体像を把握したい場合には、本章は飛ばしてもかまいません。

4.1 主なHTML要素

HTML内で利用する主な要素のうち、前章で扱わなかったものを紹介します¹⁾。

4.1.1 画像 : img

適当な画像ファイル (image.png) を用意してください。HelloWorld.html を次のように書き換えてウェブブラウザでリロードすると、画像が表示されます。

```
<html>
<body>
<p><img src='image.png' alt='サンプル画像'></img></p>
</body>
</html>
```

画像を提示するには **img 要素** を使います。提示する画像のファイル名は「src=」ファイル名」のように指定します（「src='http://www.example.net/image.jpg」のように、ウェブ上の画像を指定することもできます）。このように、開始タグの中で指定するものを **属性** といいます（img 要素の実体は **src 属性** で設定するというわけです）。属性の値は引用符（'」または「」）で囲みます。img 要素では、画像を表示できないブラウザなどのために、**alt 属性** を設定することになっています。

この img 要素のように、内容が無い場合には、終了タグを書く代わりに、「」と書くこともできます（つまり **開始タグの最後を「>」ではなく「/>」にします**）。

4.1.2 改行 : br

p 要素などの中で強制的に改行したいときに **br 要素** を使います。改行を
としている資料がたくさんありますが（HTMLならこれでよかったのです）、XHTML（後述）では

1) HTML 要素の全てを本書で扱うことはできません。JavaScript を使って絵を描くことができる **canvas 要素** など、便利なものが他にたくさんあります。

とするのが正しい書き方です。

4.1.3 整形済みテキスト : pre

概念

pre 要素中の改行や空白はそのまま表示されます。
そのため、pre 要素はソースコードなどを記述する際によく使われます。

実装

```
<pre>pre 要素中の改行や空白はそのまま表示されます。  
そのため、pre 要素はソースコードなどを記述する際によく使われます。 </pre>
```

(ソースコードであることを示すには、code 要素を使います。)

4.1.4 引用 : blockquote, q

概念

テキストをまとめた引用文 (block-level quotation) として定義するには、blockquote 要素を使用する。(益子貴寛『Web 標準の教科書』)

実装

```
<blockquote>  
<p>テキストをまとめた引用文(block-level quotation)として定義するには、blockquote  
要素を使用する。(益子貴寛『Web標準の教科書』)</p>  
</blockquote>
```

概念

“比較的短い文章をインラインで引用する場合は q 要素を使用する” ことになっています。

実装

```
<p><q>比較的短い文章をインラインで引用する場合はq要素を使用する</q>ことになっていま  
す。 </p>
```

(q 要素に対応したウェブブラウザなら引用符が自動的に付きます。)

4.1.5 強調 : em , strong

強調には em 要素と strong 要素があります。多くのブラウザでは em 要素は斜体で、strong 要素は太字で表示されます。より強く強調したいときに strong 要素を使うのが正しい使い方ですが、日本語の斜体は美しくないなので、日本語に限っては strong 要素のみを使

CAPTER 5

データの送信

本章では、クライアントからサーバにデータを送信する方法を学びます（図 5.1）。受信する方法は 6.2 節 (p.52) で説明します。

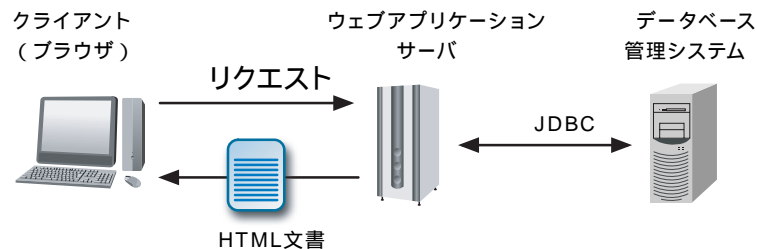


図 5.1 本章で学ぶこと：データ送信の送信方法

❶ 付録 C を参考に XAMPP をインストールしてから先に進んでください。

5.1 サーバの準備

データの送信方法を試すためにはまず、受信する側（図 5.1 のウェブアプリケーションサーバ）を用意しなければなりません。次のようなファイル RequestedParameters.jsp を \$CATALINA_HOME/webapps/ROOT に置いてください。CATALINA_HOME というのは Tomcat をインストールしたディレクトリのことで、Windows では「C:\xampp\tomcat」あるいは「C:\Program Files\xampp\tomcat」、GNU/Linux では「/opt/lampp/tomcat」、Mac OS X では「/Applications/xampp/xamppfiles/tomcat」になります¹⁾。

受信側のプログラムの書き方を学ぶのは次章以降なので、この段階では意味がわからなくてもかまいません。

```
<?xml version="1.0" encoding="UTF-8" ?>
<%@ page language="java" contentType="text/html; charset=UTF-8"
import="java.util.*"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

1) Windows では \$CATALINA_HOME ではなく %CATALINA_HOME% のように参照するのですが、本書では特に区別しません。

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Requested Parameters</title>
</head>
<body>
<table>
<tr><th>name</th><th>value</th></tr><%
  for(Enumeration e=request.getParameterNames();e.hasMoreElements();){
    String name=(String)e.nextElement();
    for(String value:request.getParameterValues(name)){%>
      <tr>
        <td><c:out value="<%=new String(name.getBytes("ISO8859_1"),"UTF-8") %>" /></td>
        <td><c:out value="<%=new String(value.getBytes("ISO8859_1"),"UTF-8") %>" /></td>
      </tr><%>
    }
  }%>
</table>
</body>
</html>

```

5.1.1 サーバの起動

Tomcat を起動して, \$CATALINA_HOME/webapps/ROOT に置いた RequestedParameters.jsp を利用可能にします。起動方法は次の通りです。

Windows : %CATALINA_HOME%\bin\startup.bat を管理者権限で実行します。Windows ファイアウォールの警告に対して、「ブロックを解除する」をクリックします。(コントロールパネルで設定することもできます)。うまく動かないときは、コマンドプロン



図 5.2 Tomcat の起動画面

CAPTER 6

ダイナミックなページ生成

本章ではダイナミックなページ生成について説明します。ダイナミックにページを生成することは、ウェブアプリケーションサーバの基本的な機能です(図 6.1)。

❗ 本章を読むには Java の知識が必要です。付録 A を参照してください。

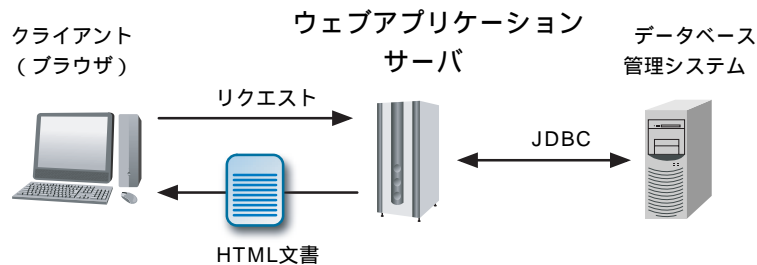


図 6.1 本章で学ぶこと：ダイナミックなページ生成方法

ダイナミックなページ生成とは、具体的にはどういうことを意味するのでしょうか。ダイナミックなページ生成の反対がスタティックなページ生成です。これは、4.4 節 (p.33) で見たように、あらかじめ作成しておいたページを提示することを言います。それに対して、要求があったその時にページを生成するのがダイナミックなページ生成です。

もう少し詳しく説明しましょう。3.2 節 (p.17) で書いた HelloWorld.html は次のようなものでした(これは妥当な HTML 文書ではありませんが、ここでは気にしません)。

```
<html>
<body>
Hello World!
</body>
</html>
```

ダイナミックにこのページを生成するというのは、次のようなプログラムを実行するということです(これは説明のための擬似コードだと思ってください)。

```
printf "<html>\n"
printf "<body>\n"
printf "Hello World!\n"
printf "</body>\n"
printf "</html>\n"
```

しかしこれでは手間が増えただけのように見えます。もう少し利用価値のわかる例を考えましょう。たとえば、「1 から 100 までの数値を箇条書きにする」という場合はどうで

しょうか。次のようなコードで実現できますが（これも説明のための擬似コードだと思ってください）、li 要素を 100 個書くのに比べればはるかに簡単です。

```
printf "<ul>";
i=1;
while [ $i -le 100 ]
do
  printf "<li>%d</li>" $i;
  i=$((i+1));
done
printf "</ul>";
```

このままでは、ダイナミックに生成する必要のない要素もプログラムで生成しなければなりません。たとえば、4.2.1 項 (p. 23) で紹介した各種宣言を、わざわざ次のように書かなければならないのでしょうか¹⁾。

```
printf '<?xml version="1.0" encoding="UTF-8" ?>';
printf '<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN";';
printf ' "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">';
printf '<html xmlns="http://www.w3.org/1999/xhtml">';
...
(面倒)
```

実行前から決まっている部分はそのまま書いて、ダイナミックに生成したいところだけ、プログラムで生成するようにしたいものです。それを可能にするのが、本章で紹介する **JSP** (JavaServer Pages) です。JSP のファイルは、一見ふつうのウェブページですが、ところどころ Java のコード（あるいは特殊なタグ）が埋め込まれます。

6.1 JSP: HTML と Java の融合

6.1.1 JSP 版 Hello World

JSP 版 HelloWorld は次のようになります。

```
<html>
<body>
<p><% out.println("Hello World!"); %></p>
</body>
</html>
```

もちろん、「<p>Hello World!</p>」でいいのですが、それでは単なる HTML 文書になってしまうため、あえて Java のプログラムを書いています。このように、<%と%>の間に Java のプログラムを書いたものを **スクリプトレット** と呼びます²⁾。

- 1) このコードは **シェルスクリプト** なので実行できます。二重引用符と一重引用符で意味が変わるのはよくあることです。
- 2) スクリプトレットには、本文中で用いた「<% Java のコード %>」という書き方 (**JSP シンタックス**) と「<jsp:scriptlet> Java のコード </jsp:scriptlet>」という書き方 (**XML シンタックス**) があります。

CAPTER 7

データベースの操作 1

この章ではデータベース管理システム (Database Management System, DBMS) の操作方法を解説します。

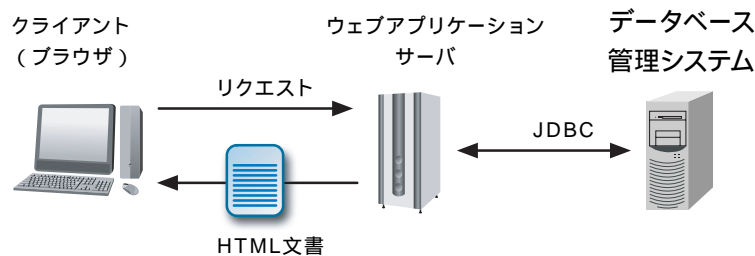


図 7.1 本章で学ぶこと：DBMS の操作方法

7.1 DBMS の必要性

そもそも、なぜ DBMS が必要なのでしょう。

オンライン・ショッピング・サイトを運営することを考えてみてください。顧客がログインしようとする、その顧客情報がデータベースに問い合わせられます。

顧客には商品のリストを提示しなければなりません、リストを HTML ファイルに書いておくわけにはいきません。扱う商品が変わるたびに HTML ファイルを書き換えるのは大変だからです。そこで、商品情報をデータベースに格納しておいて、そこからデータを取り出して、ページをダイナミックに生成することになります。このような**データの独立性**によって、システムが変化に対して柔軟になります。

顧客が商品を購入すると、システムは記録されている在庫数を書き換えなければなりません。このようなデータを、ふつうのファイルに記録するとどうなるでしょうか。膨大な商品データを格納したファイル中の目的のデータに高速アクセスするためには、特別な方法を用意しなければなりません。複数の顧客が商品を購入しようとするとうなるでしょう。一つのファイルに同時に書き込もうとするとおかしなことになりますから、一人がアクセスしている間はファイル全体をロックする（書き込みあるいは読み書きを禁止する）のがふつうでしょう。それでは多くの顧客に対応できませんから、ファイルの一部だけをロックするような、精巧な**排他制御**が必要になります。

サイトのデータへのアクセス権は、人によって細かく変わります。商品情報のうち、顧客がアクセスできる部分と運営者がアクセスできる部分とは当然異なります。細かいアクセス制御を、OS が持つファイルのアクセス権管理機能で実装しようとするのは非現実的です。そのため、別に**アクセス管理機能**を実装しなければなりません。

以上のようなデータの独立性や排他制御機能、アクセス管理を提供するのが DBMS です。

データベースにはさまざまな形式がありますが、本書で利用するのはリレーショナル・データベース (Relational Database, **RDB**) と呼ばれるものです。RDB の管理システムが **RDBMS** (Relational Database Management System) です。

7.2 リレーショナル・データベース管理システム

RDBMS がどのようにデータを管理しているかを簡単に描くと、図 7.2 のようになります。RDBMS が 1 つ以上のデータベースを持ち、各データベースはまた 1 つ以上のテーブル (表) を持ちます¹⁾。表計算ソフトに慣れている人は、RDBMS が表計算ソフトに、1 枚のシートがデータベースに、シートに書かれた表 (複数) がテーブルに対応すると思ってもらえればいいでしょう。

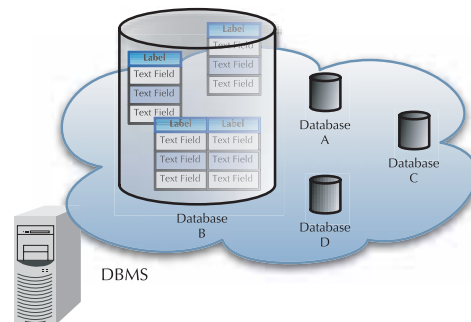


図 7.2 RDBMS の概念図

本書で用いる RDBMS は MySQL です。

MySQL を選択した理由は、自由に無料で使うことができること、広く普及しているためウェブ上の情報や書籍がたくさんあることです。レンタル・サーバにウェブアプリを置く場合、RDBMS の選択肢が MySQL のみということもよくあります。

無料で使えるとは言っても、さまざまな機能が実装されているので、使いこなすまでにはかなりの経験が必要でしょう。MySQL の機能を本書ですべて紹介することはできませんので、適宜リファレンス・マニュアルを参照してください (p. 106 を参照)。

1) これは MySQL の用語であって、RDBMS によっては別の用語が使われる場合があります。たとえば Oracle の場合、図 7.2 の意味で MySQL の「データベース」に概念的に近いのは「スキーマ」です。逆に、Oracle の「データベース」は MySQL の「データベース領域 (データベース全体の物理領域)」です。

CAPTER 8

データベースの操作2

前章に引き続きデータベースについて学びます。ウェブアプリの全体像をまず把握したいという方は、この章はとばしてください。

8.1 SELECT 文の詳細

7.4.2 項 (p.65) で紹介した SELECT 文は次のようなものでした。

```
SELECT 取り出したいもの
FROM 対象テーブル
WHERE 対象を限定する条件
```

SELECT 文にはほかにもさまざまな機能があります。表 8.2(p. 80) のデータを使って説明しましょう。

取り出したいデータに **DISTINCT** を付けると、結果から重複を取り除くことができます。

```
mysql> SELECT DISTINCT publisher FROM books;
+-----+
| publisher |
+-----+
| アスキー |
| ピアソンエデュケーション |
+-----+
```

GROUP BY 節によって、カラムの値が同じ行をまとめて処理することができます。集約関数 (表 8.10 p. 93) とともに使うことが多いでしょう。次のよう書けば、テーブル books のすべてのデータから、publisher ごとに平均価格を計算できます。

```
mysql> SELECT publisher, AVG(price)
-> FROM books
-> GROUP BY publisher;
+-----+-----+
| publisher | AVG(price) |
+-----+-----+
| アスキー | 5895.0000 |
| ピアソンエデュケーション | 4830.0000 |
+-----+-----+
```

HAVING 節には、GROUP BY 節で生成された結果に対する条件を記述します。条件の

書き方は WHERE 節 (7.4.2 項 p.65) と同じです。次のように書けば、平均価格が 5,000 より大きいものに絞り込みます。

```
mysql> SELECT publisher, AVG(price) AS x
-> FROM books
-> GROUP BY (publisher)
-> HAVING x>5000;
+-----+-----+
| publisher | x          |
+-----+-----+
| アスキー | 5895.0000 |
+-----+-----+
```

ORDER BY 節によって、結果を並び替えることができます (「ORDER BY price DESC」とすれば降順になります)。

```
mysql> SELECT title,price
-> FROM books
-> ORDER BY price;
+-----+-----+
| title                | price |
+-----+-----+
| フリーソフトウェアと自由な社会 | 3360 |
| ハッカーズ大辞典         | 3990 |
| 計算機プログラムの構造と解釈 | 4830 |
| プログラミング作法       | 5940 |
| The Art of Computer Programming Volume 1 | 10290 |
+-----+-----+
```

8.1.1 SELECT 文の構文

SELECT 文の構文は正確には次のようになります。[] は省略可能であることを意味します。

```
SELECT
  [ALL | DISTINCT | DISTINCTROW ]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
  select_expr, ...
  [FROM table_references]
  [WHERE where_condition]
  [GROUP BY {col_name | expr | position}
  [ASC | DESC], ... [WITH ROLLUP]]
  [HAVING where_condition]
  [ORDER BY {col_name | expr | position}
  [ASC | DESC], ...]
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]
  [PROCEDURE procedure_name(argument_list)]
  [INTO OUTFILE 'file_name' export_options
  | INTO DUMPFILE 'file_name'
  | INTO @var_name [, @var_name]]
```



データベースへの接続

広く使われているプログラミング言語には、RDBMS へのアクセス手段（インターフェース）が用意されています（図 9.1）。もちろん、本書で使っているプログラミング言語である Java にも **JDBC** (Java DataBase Connectivity) というインターフェースがあります¹⁾。本章では JDBC の使い方を説明します。

9.1 JDBC: Java とデータベースの架け橋

Java と RDBMS をつなぐ JDBC は RDBMS ごとに用意されていて、MySQL 用のものは **MySQL Connector/J** です。

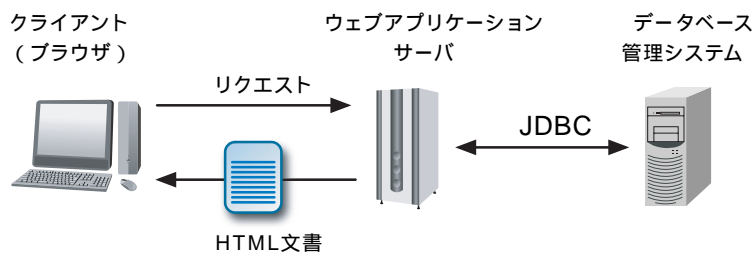


図 9.1 本章で学ぶこと：JDBC の使い方

9.1.1 MySQL Connector/J

まず、MySQL Connector/J がインストールされていることを確認しましょう。
%CATALINA_HOME%/common/lib に mysql-connector-java-バージョン番号-bin.jar というファイルがあれば MySQL Connector/J を利用するための準備は完了しています。ファイルが場合は、C.1.6 項 (p. 187) を参考に MySQL Connector/J をインストールしてください。

9.1.2 RDBMS のアクセス権限

ウェブアプリから RDBMS にアクセスする際の権限を確認します。権限の設定方法は 7.5 節 (p.68) を参照してください。MySQL は権限をカラム単位で細かく設定することができますが、本書ではデータベース単位で設定します。データベースへのアクセス権限は、

1) 広く普及しているものとしては、JDBC のほかに **ODBC** (Open DataBase Connectivity) があります。

データベース mysql のテーブル db に格納されています。ここで使うデータベース mydb へのユーザ test のアクセス権限は、設定されているなら次のように確認できます（7.6 節 p. 69 で紹介した phpMyAdmin で確認することもできます）。

```
mysql> SELECT * FROM mysql.db WHERE Db='mydb'\G
***** 1. row *****
      Host: localhost
        Db: mydb
       User: test
Select_priv: Y
Insert_priv: Y
Update_priv: Y
Delete_priv: Y
Create_priv: N
  Drop_priv: N
   Grant_priv: N
References_priv: N
   Index_priv: N
   Alter_priv: N
Create_tmp_table_priv: N
 Lock_tables_priv: N
 Create_view_priv: N
  Show_view_priv: N
Create_routine_priv: N
Alter_routine_priv: N
   Execute_priv: Y
```

ユーザ mydb は localhost からアクセスする際に、データベース mydb に対して、SELECT と INSERT, UPDATE, DELETE, EXECUTE を実行できることがわかります。

アクセス権限の設定はとても重要です。もしウェブアプリにセキュリティ上の欠陥があったとしても、アクセス権が必要最小限になっていれば、システムへの被害を最小限にすることができます²⁾。逆に、ウェブアプリが管理者として RDBMS にアクセスするようにしていると、その脆弱性をつかれたときに、データベースの全データにアクセスされてしまいます。

ウェブアプリから MySQL にアクセスする際には、専用のユーザを作成し、そのユーザとしてアクセスするようにしてください。

9.2 JDBC の利用

9.2.1 SQL の実行

最初の例として、データベース mydb のテーブル samples に対して、UPDATE 文を実行してみましょう。次のような Jdbc.jsp を作成してください。（JSTL が導入されている必

2) テーブルに対する SELECT でさえも権限が大きすぎるという場合には、適切なビューを作成し、そのビューだけに対するアクセス権を与えるようにします（ビューは本書では扱いません。リファレンス・マニュアルの“Views”の章を参照してください）。

CAPTER 10

ウェブアプリの例

JSP と MySQL , JDBC が使えるようになれば , 簡単なウェブアプリなら作ることができます . ここでは例として図 10.2 のような郵便番号検索システムを作成します .

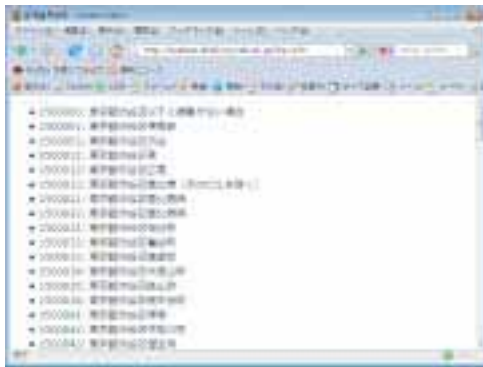


図 10.1 URL による郵便番号検索

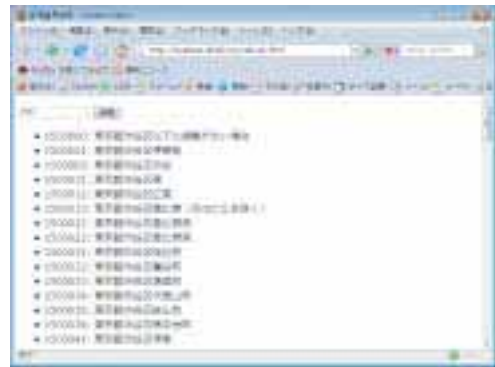


図 10.2 フォームからの郵便番号検索

まず , `http://localhost:8080/mydb/zip.jsp?zip=150` のような URL で郵便番号を検索できるようにします (図 10.1) . それができたら , フォームから検索できるように改良しましょう (図 10.2) .

10.1 郵便番号検索

まず , データを用意します .

日本の郵便番号のデータは , ゆうびんホームページ (`http://www.post.japanpost.jp/zipcode/download.html`) からダウンロードできます . ここで , 「住所の郵便番号—読み仮名データの促音・拗音を小書きで表記しないもの」の全国一括 (`ken_all.lzh`) と 「事業所の個別郵便番号」の最新全データ (`jigyosyo.lzh`) をダウンロードします .

展開してできる CSV ファイルの文字コードは Windows-31J です . 本書では , 文字コードは UTF-8 に統一することにしてるので , 一度テキストエディタで開いて , UTF-8N (BOM なしの UTF-8) で保存し直します . ファイル名は `KEN_ALL_UTF8.CSV` と `JIGYOSYO_UTF8.CSV` とします (これは文字コードを変換する練習です . Windows-31J のままで進めることも可能です) .

拡張子からもわかるように , これらは **CSV 形式** , つまりフィールドがカンマで区切られたファイルです (テキストエディタで開くとわかります) . ここで必要なフィールドは表

10.1 のとおりです (フィールドの定義はゆうびんホームページにあります)。たとえば、住所の郵便番号ファイルの 7 番目と大口事業所等個別番号ファイルの 4 番目のフィールドは「都道府県名」です。

表 10.1 郵便番号データの要素

カラム名	住所の郵便番号		大口事業所等個別番号	
	順番	内容	順番	内容
jis	1	全国地方公共団体コード	1	大口事業所等の所在地の JIS コード
old_zip	2	(旧) 郵便番号 (5 桁)	9	現行郵便番号
zip	3	郵便番号 (7 桁)	8	大口事業所等個別番号
addr1_ruby	4	都道府県名 (カナ)		
addr2_ruby	5	市区町村名 (カナ)		
addr3_ruby	6	町域名 (カナ)		
addr1	7	都道府県名	4	都道府県名
addr2	8	市区町村名	5	市区町村名
addr3	9	町域名	6	町域名
addr4			7	小字名, 丁目, 番地等
establishment_ruby			2	大口事業所等名 (カナ)
establishment			3	大口事業所等名 (漢字)

データベース mydb に次のようなテーブルを作成しましょう。

```
CREATE TABLE zips(
  jis CHAR(10) DEFAULT '' NOT NULL,
  old_zip CHAR(5) NOT NULL,
  zip CHAR(7) NOT NULL,
  addr1_ruby VARCHAR(10) DEFAULT '' NOT NULL,
  addr2_ruby TEXT NOT NULL,
  addr3_ruby TEXT NOT NULL,
  addr1 VARCHAR(10) DEFAULT '' NOT NULL,
  addr2 TEXT NOT NULL,
  addr3 TEXT NOT NULL,
  addr4 TEXT NOT NULL,
  establishment_ruby TEXT NOT NULL,
  establishment TEXT NOT NULL,
  KEY zip_idx (zip)
) DEFAULT CHARACTER SET utf8;
```

作成したテーブル zips にデータをロードします (ファイルの文字コードが UTF-8 なので、まずその設定をします)。カラム名を順番に書くことで、フィールドがテーブルのどのカラムに対応するかを指定します。LOAD 文は次のようになります。

```
SET NAMES utf8;

LOAD DATA LOCAL INFILE "KEN_ALL_UTF8.CSV" INTO TABLE zips
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
(jis,old_zip,zip,addr1_ruby,addr2_ruby,addr3_ruby,addr1,addr2,addr3);

LOAD DATA LOCAL INFILE "JIGYOSYO_UTF8.CSV" INTO TABLE zips
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
(jis,establishment_ruby,establishment,addr1,addr2,addr3,addr4,zip,old_zip);
```

CAPTER 11

Model, View, Controller

ウェブアプリを model, view, controller という三つの要素で組み立てる方法を紹介します。一見複雑に見えますが、この枠組みで作ると、後の修正や拡張、部品の再利用が容易になります。

11.1 MVC とは何か、どう実装するのか

11.1.1 Monolithic JSP

10 章 (p.119) で作成したウェブアプリのサーバ側は、ただ一つの JSP (zip.jsp) からできていました。zip.jsp が担っていたのは次の処理です。

1. クライアントからデータを受信する
2. データベースへ問い合わせる
3. 結果をクライアントに返す

このように、処理のすべてを担うような JSP は、Monolithic JSP と呼ばれ、ウェブアプリ開発のアンチパターン（よい設計であるデザインパターンの対極）とされています（図 11.1）[47]。アプリケーションが複雑になると、JSP は手に負えないくらい複雑になってしまうからです。そうならないために、MVC というデザインパターンに沿ってアプリケーションを実装します。



図 11.1 Monolithic JSP

11.1.2 Model-View-Controller

ウェブアプリにおける MVC とは、おおざっぱに言えば、アプリケーションの各役割を次のように分担することです（図 11.2）。

Controller：クライアントからのデータを受け取ります。実装は Servlet です。

Model：メイン・タスク（ビジネスロジック）を担います。実装は JavaBeans です¹⁾。

View : クライアントに返す画面を生成します . 実装は JSP です .

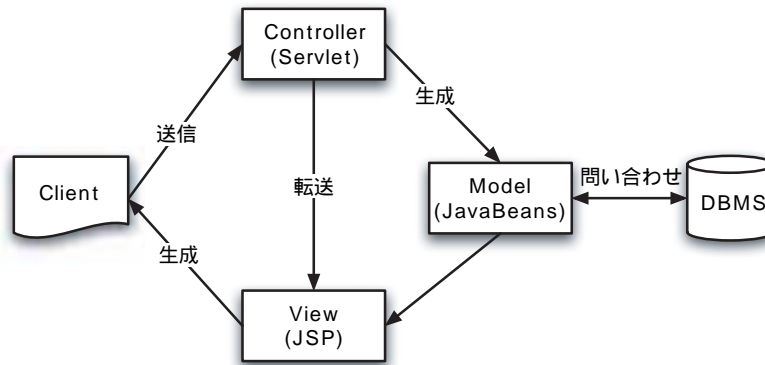


図 11.2 Model-View-Controller

11.2 郵便番号検索の MVC による実装

先の郵便番号検索システムを MVC で実装すると、各コンポーネントの動作は次のようになります (図 11.3)。

1. Client: zip.Controller にデータ (code=194) を送信する .
2. Controller: zip.Model を生成する .
3. Controller: zip.Model のパラメータを設定する (code="194") .
4. Controller: zip.Model のメイン・タスク (検索) を実行させる .
5. Model: データベースに問い合わせる . (SELECT 文の実行)
6. Controller: zip.Model を一時保管する .
7. Controller: View.jsp に移行する .
8. View: 保管された zip.Model を取り出す .
9. View: zip.Model から情報を取り出しながらページを生成する .

これだけみると非常に複雑になってしまったように思うかもしれませんが、しかし、こうすることで、実装がとても楽になるのです。アプリケーションが複雑になっても、基本的な枠組みはこれ以上複雑にはなりません。

1) Model は JavaBeans ではなく、EJB (Enterprise JavaBeans) で作られることも多いです。しかし、EJB はかなり複雑なため、本書では扱いません (興味のある方は、EJB 3.0 に対応した資料にあたってください)。いずれにしても、Model をウェブアプリから切り離すことで、別のアプリケーションで再利用することができるようになります。

CAPTER 12

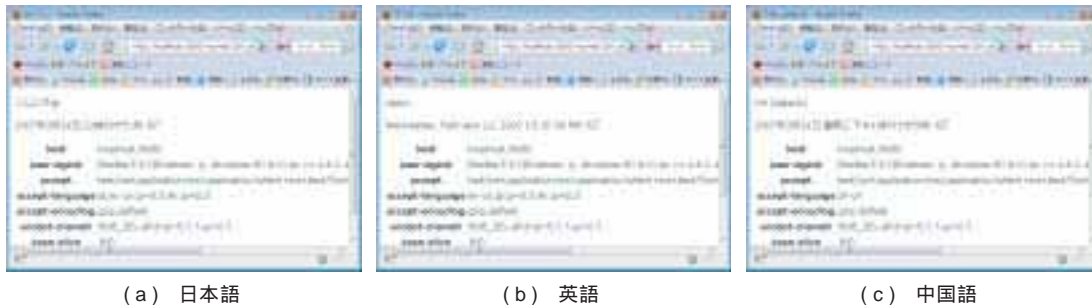
国際化

国際化 (internationalization, 長いため間の 18 文字を略して **i18n** と書きます [25]) とは、アプリケーションをさまざまな言語や地域に対応可能にすることをいいます。国際化されたアプリケーションを、実際に特定の言語や地域に対応させるのが地域化 (localization, **l10n**) 1つのアプリケーションで複数の言語を扱えるようにするのが**多言語化** (multilingualization, **m17n**) です。

本章ではウェブアプリを国際化・地域化する基本的な方法を紹介します。

12.1 国際化とは

i18n-sample.jsp という JSP があります (コードは後で紹介します)。Firefox でこの JSP を閲覧すると、図 12.1(a) のようになります。「ツール ⇒ オプション ⇒ 詳細 ⇒ 一般タブ ⇒ 言語設定」で英語/米国 [en-us] の優先順位を最高にしてから閲覧すると、図 12.1(b) のようになります。日本語と英語を削除し、それら以外の適当な言語を追加して閲覧すると、図 12.1(c) のようになります。つまり、JSP 自体は変えることなく、提示する情報をクライアントの言語設定にあわせることができます。i18n-sample.jsp は**国際化**され、日本語と英語に**地域化**されているのです。



(a) 日本語

(b) 英語

(c) 中国語

図 12.1 i18n-sample.jsp の閲覧結果

地域化は、自然言語の部分をそのまま翻訳し、それを利用可能にするだけでは不十分です。たとえば、日付の表記は、日本ならば「1981年1月28日」となるところが、USAでは「January 28, 1981」、フランスでは「28 janvier 1981」となります。つまり、地域化のためには言語だけでなく表記規則にも対応しなければなりません。地域化は**ロケール**に対して行われます。ロケールとは、文化的、地理的、政治的な地域と言語の組み合わせです。

国際化と地域化は、一見とても面倒に見える要求ですが、そのための仕組みはすでに用意されており、それに合うようにウェブアプリを実装する簡単に実現できます。トップページで言語を選べないウェブサイトやウェブアプリは、何かが間違っています。



CプログラマのためのJava

C言語を知っている人向けに、ウェブアプリ作成のために最低限知っておかなければならないJavaの知識をまとめます。

A.1 オブジェクト

オブジェクトとは、**属性と操作**を一つにまとめたものです¹⁾。属性の実装は `int` や `double` などの基本型あるいはオブジェクト型の**変数**、操作の実装は**メソッド**（C言語の関数のようなもの）です。

A.1.1 クラス

オブジェクトを定義するのが**クラス**です。例として、`firstName`、`lastName` という変数と `run` というメソッドを持つクラス `Person` を考えましょう（クラス名の先頭は大文字、変数名・メソッド名の先頭は小文字にするのが一般的です）。`Person` などというありふれた名前のクラスをそのまま使うと名前が衝突するおそれがあるので、`mypackage` というパッケージを用意し、その中で `Person` を定義することにします。クラスについて説明するとき、図 A.1 のような **UML クラス図** を書くのが一般的です。

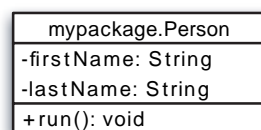


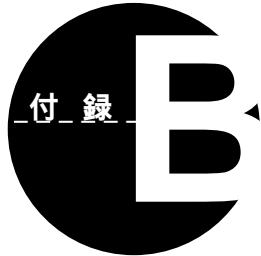
図 A.1 Person の UML クラス図

変数名やメソッド名の前に付いている記号（「-」や「+」）は**可視性**（visibility）を表します（表 A.1）。

Eclipse 上での実装

Eclipse にはクラスの骨格を生成する仕組みが備わっています。それを使ってクラスを実装しましょう。

1) C言語の構造体は、変数を一つにまとめたものでした。



文字コード

私は宣言する．もしあなたが 21 世紀において仕事をしているプログラマであり，キャラクタ，キャラクタセット，エンコーディング，Unicode の基本について知らないのであれば，私はあなたをひっ捕まえて，潜水艦で 6 か月のたまねぎ剥きの刑に処する．—Joel Spolsky[44]

B.1 文字コードとは

コンピュータが扱えるデータは，突き詰めればバイト列（ビット列）だけです．人間が何気なく扱っている「文字（キャラクタ）」も，コンピュータで扱うためにはバイト列で表さなければなりません．文字をバイト列で表す方法はいろいろ考えられますが，最も一般的なのは文字コードを利用する方法です．

文字コードとは，文字集合とその符号化方式の組のことです．符号化文字集合とも呼ばれます．文字集合は文字通り文字の集合¹⁾，符号化方式は文字集合内の文字への番号の振

表 B.1 主要な文字コード

文字コード	文字集合	符号化方式	補足
ASCII			最も普及している文字コード．7 ビットの空間で 128 文字を表現する．制御文字と空白を除くと 94 文字．(図 6.7 p. 51)
ISO-8859-1			ASCII (94 文字) と西ヨーロッパ文字 (96 文字) を合わせたもの．Latin 1 とも呼ばれる．
JIS X 0201			ASCII (一部変更) にいわゆる「半角カナ」を追加したもの．特殊文字を除くと 158 文字 (空白を含む)．
ISO-2022-JP	JIS X 0201+ 208	ISO-2022	JIS X 0201 の半角カナは含まない．日本語のメールではこの文字コードがよく用いられる．ISO の規格ではない．JIS コードとも呼ばれるが，JIS 規格でもない．
EUC-JP	JIS X 0201+ 0208+ 0212	ISO-2022	Unix などよく使われていた文字コード．
Shift_JIS	JIS X 0201+ 0208	Shift_JIS	名称に JIS とあるが，JIS 規格ではない．
Windows-31J	JIS X 0201+ 0208+ 特殊文字	Shift_JIS	日本の PC のデファクト．間違っても Shift_JIS と呼ばれることが多い．
UTF-8	UCS	UTF-8	Unicode の全ての文字を扱えるが，漢字は 1 文字あたり 3 バイト以上必要．
UTF-16	UCS	UTF-16	2 バイトあるいは 4 バイトで 1 文字を表す．

1) 集合とは言っても番号を振って定義されているものが多いです．そういう意味では文字コードだとも言えます．このあたりは厳密に使い分けられているわけではありません．たとえば，ASCII は文字コードの意味でも文字集合の意味でも使われます．



開発環境の構築

ここでは、本書のための環境を構築する方法を紹介します。インストールするソフトウェアは次の通りです。

- Mozilla Firefox
- Java SE Development Kit (JDK)
- Apache Tomcat
- MySQL
- Eclipse + Web Tools Platform (WTP)

MySQL は、**XAMPP** というソフトウェア・パッケージの一部としてインストールします。XAMPP は、MySQL の他に標準的なウェブサーバーである **Apache HTTP Server** や **PHP** など、ウェブ開発に役立つツールを収録したパッケージです。本書の内容との関連で言えば、ブラウザから MySQL を操作できる **phpMyAdmin** (p. 69 を参照) が利用可能になることが、最大のメリットでしょう。

OS 別に環境構築方法を紹介します。

C.1 Windows Vista および XP

Windows Vista あるいは **XP** 上で開発環境を構築する方法を説明します¹⁾。作業は、Vista では標準ユーザとして、XP では管理者として行います²⁾。

C.1.1 Windows の設定

ファイルの拡張子を表示する

Windows では、ファイルの種類をその拡張子 (最後の「.」以降の文字列) で判別します。Windows の初期設定では、登録されている拡張子は表示しない設定になっていますが、これでは不便なので、すべての拡張子を表示するように設定を変更します。

コントロールパネル ⇒ デスクトップのカスタマイズ (XP では「デスクトップの表示とテーマ」) ⇒ フォルダオプション ⇒ 表示タブ ⇒ 「登録されている拡張子は表示しない」のチェックを外す (図 C.1)

1) スクリーン・ショットは VMware Server 1.0 上にインストールした Windows Vista RC1 でのものです (Windows XP でも動作確認はしています)。

2) Windows のユーザには、大きく分けて標準ユーザと管理者 (Administrator) があり、ソフトウェアのインストールやサーバー・プログラムの操作は管理者でなければできません。安全を考慮して、ふだんは標準ユーザとして利用し、必要なときだけ管理者になるようにすべきです。Windows Vista では、管理者権限が必要なときは、そのパスワードを入力することによって、一時的に管理者になることができます。Windows XP にはそのような機能はないため、常に管理者として利用するのも仕方ありません。

参考文献

- [1] banban@scollabo.com. 初心者のためのホームページ作り. <http://www.scollabo.com/banban/>, 2006. (本文 p. 35).
- [2] Alan Beaulieu. *Learning SQL*. O'Reilly & Associates Inc., 2005. クイープ訳. 初めてのSQL. オライリー・ジャパン, 2006. (本文 p. 86, 106).
- [3] Hans Bergsten. *JavaServer Pages*. O'Reilly & Associates, Inc., 2nd edition, 2001. 光田秀訳. JavaServer Pages. オライリー・ジャパン. 第2版, 2003. (本文 p. 52, 141).
- [4] T. Berners-Lee, R. Fielding, and L. Masinter. RFC3986: Uniform resource identifier (URI): Generic syntax. <http://www.ietf.org/rfc/rfc3986.txt>, 2005. H-Hash 訳. <http://www.studyinghttp.net/cgi-bin/rfc.cgi?3986>. (本文 p. 175).
- [5] Kernighan B.W. and Ritchie D.M. *The C Programming Language*. Prentice Hall, 2nd edition, 1988. 石田晴久訳. プログラミング言語C. 共立出版, 第2版, 1989. (本文 p. 5).
- [6] Masakatz Canada. メールアドレスに一致する正規表現. <http://www.tt.rim.or.jp/~canada/comp/cgi/tech/mailaddrmatch/>. (本文 p. 156).
- [7] Joe Celko. *Joe Celko's SQL for Smarties: Advances SQL Programming*. Academic Press, 2nd edition, 2000. 秋田昌幸訳. プログラマのためのSQL. ピアソンエデュケーション, 第2版, 2001. (p.106)
- [8] Joe Celko. *Joe Celko's SQL Puzzles and Answers*. Morgan Kaufmann Pub., 2nd edition, 2006. (本文 p. 106).
- [9] David Crane, Eric Pascarello, and Darren James. *Ajax in Action*. Manning Pubns Co., 2005. 柏原正三, 網代淳, 星睦訳. Ajax イン・アクション. インプレス, 2006. (本文 p. 123).
- [10] C&R 研究所. 超図解 SQL ハンドブック. エクスメディア, 2005. (本文 p. 77, 106).
- [11] Ian F. Darwin. *Java Cookbook*. O'Reilly & Associates Inc., 2nd edition, 2004. 宇野浩司, 豊福剛訳. Java クックブック. オライリー・ジャパン, 2002. (本文 p. 166).
- [12] C. J. Date. *An Introduction to Database Systems*. Addison-Wesley Pub., 8th edition, 2003. 藤原謙訳. データベースシステム概論. 丸善, 1997 (原書第6版の翻訳)(本文 p. 106).
- [13] Andrew Deitsch and David Czarnecki. *Java Internationalization*. O'Reilly & Associates Inc., 2001. 風間一洋訳. JAVA 国際化プログラミング. オライリー・ジャパン, 2002. (本文 p. 146).
- [14] David Ewing Duncan. *Calendar: Humanity's Epic Struggle to Determine a True and Accurate Year*. Bard, 1998. 松浦俊輔訳. 暦をつくった人々. 河出書房新社, 1998. (本文 p. 157).
- [15] R. Fielding, et al. RFC2616: Hypertext transfer protocol - http/1.1. <http://www.w3.org/Protocols/rfc2616/rfc2616.html>, 1999. H-Hash 訳. ハイパーテキスト転送プロトコル - HTTP/1.1. <http://www.studyinghttp.net/cgi-bin/rfc.cgi?2616>. (本文 p. 43).
- [16] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Pub., 3rd edition, 2003. 羽生田栄一訳. UML モデリングのエッセンス. 翔泳社, 第3版, 2005. (本文 p. 166).
- [17] Eric Freeman, Elisabeth Freeman, Kathy Sierra, and Bert Bates. *Head First Design Patterns*. O'Reilly & Associates Inc., 2004. 木下哲也, 有限会社福龍興業訳. Head First デザインパターン. オライリー・ジャパン, 2005. (本文 p. 141).
- [18] Jeffrey E. Friedl. *Mastering Regular Expressions*. O'Reilly & Associates Inc., 3rd edition, 2006. 田和勝訳. 詳説 正規表現. オライリー・ジャパン, 第2版, 2003. (本文 p. 166).
- [19] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Pub., 1995. 本位田真一 and 吉田和樹訳. オブジェクト指向における再利用のためのデザインパターン. ソフトバンククリエイティブ, 改訂版, 1999. (本文 p. 141).
- [20] GIJOE. PHP サイバーテロの技法. ソシム, 2005. (本文 p. 141).
- [21] Stephen Jay Gould. *Questioning the Millennium*. Random House Inc., 1997. 渡辺政隆訳. 暦と数の話—グールド教授の2000年問題. 早川書房, 1998. (本文 p. 156).

- [22] Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Pub., 1993. 喜連川優訳. トランザクション処理. 日経 BP 社, 2001. (本文 p.106).
- [23] Robert Hess. 色覚に障害を持っていたとしたら, あなたのサイトは見えるでしょうか? <http://www.microsoft.com/japan/msdn/columns/hess/hess10092000.aspx>, 2000. (本文 p.35).
- [24] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley Longman, 2nd edition, 2001. 野崎昭弘他訳. オートマトン言語理論 計算論 I および II. サイエンス社, 第 2 版, 2003. (本文 p.153).
- [25] I18nGuy. Origin of the abbreviation i18n. <http://www.i18nguy.com/origini18n.html>. (本文 p.142).
- [26] Klensin, J., editor. RFC2821: Simple mail transfer protocol. <ftp://ftp.isi.edu/in-notes/rfc2821.txt>, 2001. srgia 訳. <http://srgia.com/docs/rfc2821j.html>, 2005. (本文 p.156).
- [27] Donald Ervin Knuth. *The Art of Computer Programming*. Addison Wesley, 3rd edition, 1997. 有澤誠ほか訳. アスキー, 2004. (本文 p.104).
- [28] Mark A. Kolb. The expression language. <http://www-128.ibm.com/developerworks/java/library/j-jst10211.html>, 2003. 日本語訳 (<http://www-06.ibm.com/jp/developerworks/java/030411/j-jst10211.html>). (本文 p.52).
- [29] Ken Lunde. *CJKV Information Processing*. O'Reilly & Associates Inc., 1998. 小松章, 逆井克己訳. CJKV 日中韓越情報処理. オライリージャパン, 2002. (本文 p.74).
- [30] L. Masinter, et al. RFC2718: Guidelines for new URL schemes. <http://www.ietf.org/rfc/rfc2718.txt>, 1999. (本文 p.175).
- [31] Brett McLaughlin. *Head Rush Ajax*. O'Reilly & Associates Inc., 2006. 児島修訳. Head Rush Ajax. オライリー・ジャパン, 2006. (本文 p.123).
- [32] Eric A. Meyer. DOCTYPE grid. <http://meyerweb.com/eric/dom/dtype/dtype-grid.html/html>. (本文 p.24).
- [33] Peter Morville. *Ambient Findability*. O'Reilly & Associates Inc., 2005. 浅野紀予訳. アンビエント・ファインダビリティ—ウェブ, 検索, そしてコミュニケーションをめぐる旅. オライリージャパン, 2006. (本文 p.3).
- [34] Jakob Nielsen. Jakob Nielsen 博士の Alertbox. <http://www.usability.gr.jp/alertbox/>. (本文 p.35).
- [35] Jakob Nielsen. ヤコブ・ニールセンの Alertbox —そのデザイン, 間違ってます. RBB PRESS, 2006. (本文 p.35).
- [36] Donald A. Norman. *The Psychology of Everyday Things*. Basic Books, 1988. 野島久雄訳. 誰のためのデザイン? —認知科学者のデザイン原論. 新曜社, 1990. (本文 p.35).
- [37] Peter Norvig. Teach yourself programming in ten years. <http://www.norvig.com/21-days.html>. yomoyomo, 竹中明夫訳. プログラミングを独習するには 10 年かかる. <http://www.yamdas.org/column/technique/21-daysj.html>, 2004. (本文 p.iii).
- [38] Morville P. and Rosenfeld L. *Information Architecture for the World Wide Web*. O'Reilly & Associates Inc., 3rd edition, 2006. ソシオメディア株式会社訳. Web 情報アーキテクチャ—最適なサイト構築のための論理的アプローチ. オライリージャパン, 第 2 版, 2003. (本文 p.3, 35).
- [39] Arnold Robbins and Nelson H.F. Beebe. *Classic Shell Scripting*. O'Reilly Media, 2005. 日向あおい訳. 詳解シェルスクリプト. オライリージャパン, 2006. (本文 p.197).
- [40] Dave Shea and Molly E. Holzschlag. *The Zen of CSS Design: Visual Enlightenment for the Web*. Peachpit Press, 2005. 森本真吾訳. CSS Zen Garden Book. 毎日コミュニケーションズ, 2006. (本文 p.35).
- [41] Kathy Sierra, Bryan Basham, and Bert Bates. *Head First Servlets & JSP*. O'Reilly & Associates Inc., 2004. (本文 p.141).
- [42] Kathy Sierra and Bert Bates. *Head First Java*. O'Reilly & Associates Inc., 2nd edition, 2005. 夏目大訳. オライリージャパン, 第 2 版, 2006. (本文 p.166).
- [43] Ishida So. RFC 日本語版リスト. <http://www5d.biglobe.ne.jp/~stssk/rfcjlist.html>, 2006. (本文 p.44).
- [44] Joel Spolsky. *Joel on Software*. Apress, 2004. 青木靖訳. Joel on Software. オーム社, 2005. (本文 p.167).
- [45] Sun Microsystems, Inc. Java 国際化 FAQ. http://java.sun.com/javase/technologies/core/basic/intl/faq_ja.jsp. (本文 p.146).
- [46] Sun Microsystems, Inc. JavaServer Pages (jsp) v2.0 syntax reference. <http://java.sun.com/produ>

- cts/jsp/syntax/2.0/syntaxref20.html. (本文 p.47, 52).
- [47] Bruce Tate. *Bitter Java*. Manning Pubns Co., 2002. トップスタジオ訳. サーバサイド Java アンチパターン. 日経 BP 社, 2003. (本文 p.128).
- [48] Jenifer Tidwell. *Designing Interfaces*. O'Reilly & Associates Inc., 2005. 浅野紀予訳. デザイニング・インターフェース—パターンによる実践的インタラクションデザイン. オライリー・ジャパン, 2007. (本文 p.35).
- [49] W3C. Cascading Style Sheets, level 2 specification. <http://www.w3.org/TR/REC-CSS2/cover.html>, 1998. 日本語訳: http://www.y-adagio.com/public/standards/tr_css2/about.html. (本文 p.30).
- [50] W3C. Web content accessibility guidelines 1.0. <http://www.w3.org/TR/WAI-WEBCONTENT/>, 1999. Miki Ofuji 訳. ウェブコンテンツ・アクセシビリティ・ガイドライン 1.0. <http://www.zspc.com/documents/wcag10/>. (本文 p.35).
- [51] W3C. Recommended list of DTDs. <http://www.w3.org/QA/2002/04/valid-dtd-list.html>, 2002. (本文 p.24).
- [52] Robin Williams. *The Non-Designer's Design Book*. Peachpit Pr., 2nd edition, 2003. 吉川典秀. ノンデザイナーズ・デザインブック. 毎日コミュニケーションズ, 第2版, 2004. (本文 p.35).
- [53] W3C HTML working group. XHTML 1.0 the extensible hypertext markup language. <http://www.w3.org/TR/2000/REC-xhtml1-20000126/>, 2000. ども猫本舗訳. XHTML 1.0: 拡張可能ハイパーテキストマークアップ言語. 第1版. <http://www.doraneko.org/webauth/xhtml10/20000126/0verview.html>. (本文 p.35).
- [54] Satoh Yoshiyuki. ジョーク RFC. <http://www.imasy.or.jp/~yotti/rfc-joke.html>, 2005. (本文 p.44).
- [55] Jeremy D. Zawodny and Derek J. Balling. *High Performance MySQL: Optimization, Backups, Replication and Balancing*. O'Reilly & Associates Inc., 2004. 林秀幸訳. 実践ハイパフォーマンス MySQL. オライリージャパン, 2004. (本文 p.106).
- [56] きしだなおき. 創る Java—NetBeans でつくって学ぶ Java GUI & WEB アプリケーション. 毎日コミュニケーションズ, 2005. (本文 p.8).
- [57] 益子貴寛. Web 標準の教科書—XHTML と CSS でつくる“正しい”Web サイト. 秀和システム, 2005. (本文 p.24, 27, 33, 35).
- [58] 益子貴寛. Web 標準の基礎と実践. <http://itpro.nikkeibp.co.jp/article/COLUMN/20060613/240726/>, 2006. (本文 p.35).
- [59] 松信嘉範. 現場で使える MySQL. 翔泳社, 2006. (本文 p.106).
- [60] 李華植. データベースパフォーマンスアップの教科書 基本原理編. 翔泳社, 2006. (本文 p.100, 106).
- [61] 大藤幹. CSS プロフェッショナル・スタイル—世界の「最先端」事例に学ぶ. 毎日コミュニケーションズ, 2005. (本文 p.35).
- [62] 藤塚勤也. 徹底比較!! MySQL エンジン. <http://www.thinkit.co.jp/free/article/0608/1/1/>, 2006. (本文 p.81).
- [63] 藤塚勤也. 徹底比較!! PostgreSQL vs MySQL. <http://www.thinkit.co.jp/free/article/0603/10/1/index.html>, 2006. (本文 p.107).
- [64] 結城浩. Java 言語で学ぶデザインパターン入門. ソフトバンククリエイティブ, 増補改訂版, 2004. (本文 p.141).
- [65] 江守賢治. 字体辞典. 三省堂, 1986. (本文 p.170).
- [66] 渡辺幸三. 業務別データベース設計のためのデータモデリング入門. 日本実業出版社, 2001. (本文 p.79).
- [67] 鈴木幸市. RDBMS 解剖学. 翔泳社, 2005. (本文 p.106).
- [68] 芝野耕司. JIS 漢字字典. 日本規格協会, 増補改訂版, 2002. (本文 p.170, 177).
- [69] 佐藤匡剛. ソースコードリーディングから学ぶ Java の設計と実装. 技術評論社, 2006. (本文 p.166).
- [70] 森山将之. Windows-31J 情報. <http://www2d.biglobe.ne.jp/~msyk/charcode/cp932/index.html>, 2003. (本文 p.168).
- [71] 小形克宏. 文字の海, ビットの舟—文字コードが私たちに問いかけるもの. <http://internet.watch.impress.co.jp/www/column/ogata/>. (本文 p.177).
- [72] 小池和夫, 府川充男, 直井靖, 永瀬唯. 漢字問題と文字コード. 太田出版, 1999. (本文 p.177).
- [73] 塩田紳二. 知りたい人のための RFC の歩き方. エーアイ出版, 1999. (本文 p.44).
- [74] 深沢千尋. 文字コード超研究. ラトルズ, 2003. (本文 p.74, 177).
- [75] 川合孝典. Java は Perl よりも比較にならないほど速い? <http://homepage3.nifty.com/hippo2000/perltips/javaperl.htm>, 2002. (本文 p.11).
- [76] 大藤幹. 詳解 HTML & XHTML & CSS 辞典. 秀和システム, 2005. (本文 p.24, 27, 35).

- [77] 長谷川恭久. Web Designer 2.0 進歩し続ける Web デザイナーの考え方. ソシム, 2005. (本文 p.35).
- [78] 高橋登史朗, 古籬一浩. Ajax 実践テクニック. 秀和システム, 2006. (本文 p.123).
- [79] 円満字二郎. 人名用漢字の戦後史. 岩波書店, 2005. (本文 p.168).
- [80] 石川博. 次世代データベースとデータマイニング—DB&DMの基礎とWeb・XML・P2Pへの適用. CQ出版, 2005. (本文 p.106).
- [81] 府川充男. 漢字問題と文字コード, 当今「漢字問題」 鄙見, pp. 53-159. 太田出版, 1999. (本文 p.170).
- [82] 風間一洋. Shift_JISのエイリアスの変更について. http://www.ingrid.org/java/i18n/encoding/shift_jis.html. (本文 p.168).
- [83] 豊島正之. JIS 漢字批判の基礎知識. <http://www.joao-roiz.jp/mtoyo/on-JCS/index.html>, 1998. (本文 p.177).
- [84] 林優子. プロとしてのデータモデリング入門. ソフトバンククリエイティブ, 2006. (本文 p.79).
- [85] 鈴木啓修. MySQL 全機能リファレンス. 技術評論社, 2004. (本文 p.106).
- [86] 神崎正英. ユニバーサルHTML/XHTML. 毎日コミュニケーションズ, 2000. (本文 p.35).
- [87] Paul Dubois. *MySQL Cookbook*. O'Reilly & Associates Inc., 2nd edition, 2006. (本文 p.106).
- [88] Anthony Molinaro. *SQL Cookbook*. O'Reilly & Associates Inc., 2005. 木下哲也, 有限会社福龍興業訳. SQL クックブック. オライリー・ジャパン, 2007. (本文 p.106).
- [89] Jonathan Gennick. *SQL Pocket Guide*. O'Reilly & Associates Inc., 2004. 林秀幸訳. SQL ハンドブック. オライリー・ジャパン, 2005. (本文 p.106).
- [90] 大垣靖男. Web アプリセキュリティ対策入門. 技術評論社, 2006.
- [91] 徳丸浩. 狙われる Web アプリケーション. <http://itpro.nikkeibp.co.jp/article/COLUMN/20070401/267074/>, 2004.
- [92] 佐藤匡剛. List インターフェイスの3つのクラスを理解する. <http://www.atmarkit.co.jp/fjava/javatips/136java026.html>, 2005.
- [93] Shelley Powers. *Learning JavaScript*. O'Reilly Media, Inc., 2006. 武舎広幸, 武舎るみ訳. 初めてのJavaScript. オライリー, 2007. (本文 p.52).
- [94] Thomas H. Cormen and Charles E. Leiserson. *Introduction to Algorithms*. MIT Press, 2001. 浅野哲夫ほか訳. アルゴリズムイントロダクション第2巻. 近代科学社, 改訂2版, 2007. (本文 p.102).
- [95] Douglas R. Hofstadter. *Metamagical Themas*. Basic Books Inc., 1985. 竹内郁雄ほか訳. メタマジック・ゲーム. 白揚社, 新装版, 2005. (本文 p.179).

- c:out 54
C# 11
CACHE *see* キャッシュ
Calc (OpenOffice.org) 84
Calendar 156
Callisto 194
canvas 要素 20
caption 要素 19
Cascading Style Sheets *see* CSS
CASE 91
CATALINA_HOME 36
catch 152
CEIL 92
CHAR 62
CHAR_LENGTH 91
Character Identifier *see* CID
characterEncoding 173
charset 24
CID 168
class 属性 28, 29
clear プロパティ 32
Client VM 51, 181
CLOB 62
Code Page 932 *see* CP932
code 要素 21
Collections 164
color プロパティ 29, 32
colspan 属性 40
Comparable 165
compare 165
CONCAT 91
Connector 要素 176
CONSTRAINT 61, 104
contentType 属性 47
content プロパティ 32
context.xml 116, 118, 126
Controller 128
controller 132
cookie 138
COS 92
COT 92
CotEditor 207
COUNT 93
CP932 4, 75
cp932 169, 173
CRC32 92
CREATE DATABASE 59
CREATE INDEX 100
CREATE TABLE 60, 81
CSS 16, 26
CSV 形式 119
CURRENT_DATE 92
CURRENT_TIME 92
CURRENT_TIMESTAMP 92
cursor プロパティ 32
cut コマンド 84
Cygwin 75
C 言語 5
- D**
- Data Tools Platform *see* DTP
Database Administrator *see* DBA
DataBase Management System *see* DBMS
dataSource 属性 116
DATE 62
DATEDIFF 92
dateStyle 属性 145
DATETIME 62
DB2 107, 118
DBA 12
DBMS 2, 10, 57
DCL 82
DDL 82
dd 要素 18
DECIMAL 62
DEFAULT 61
DEGREES 92
DELETE 68
Derby 107
DESC 61, 72
DESCRIBE 61
dhcpcd 199
DIGEST 認証 126
dir2mo 200
display プロパティ 32
DISTINCT 71
DIV 92
div 要素 30
dl 要素 18
DML 82
DOCTYPE 宣言 *see* 文書型宣言
doGet 132
doPost 132
DOUBLE 62
DROP DATABASE 60
DROP INDEX 100
DROP TABLE 61
DTP 75
dt 要素 18
DUAL 67
- E**
- Eclipse 8
インストール (GNU/Linux) 204
インストール (Mac OS X) 210
インストール (Windows) 188
日本語化 189
プロジェクト 47
ワークスペース 189
EDITOR 76
EJB 129
EL 52, 139
Emacs 179, 207
EmEditor Free 179
em ダッシュ 25
em 要素 21
enctype 属性 43
ENGINE 81
Enterprise JavaBeans *see* EJB

en ダッシュ 25
 ER 図 79
 EUC-CN 176
 EUC-JP 167, 176
 Excel 84
 EXISTS 97
 EXP 92
 EXPLAIN 99
 Expression Language
 see EL
 extension (Firefox) 180
 EXTRACT 92

F

fdisk 199
 fileUpload 52
 find 106
 Firebird 108
 FireBug 180
 Firefox 16, 179, 201
 HTML Validator 22
 Live HTTP Headers
 145
 インストール
 (GNU/Linux) 201
 インストール (Mac OS X)
 208
 インストール (Windows)
 179
 Flash 2
 FLOAT 62
 float プロパティ 32
 FLOOR 92
 fmt:formatDate 145
 fmt:message 144, 146
 fmt:param 146
 fmt:setBundle 144
 font-family プロパティ
 32
 font-size プロパティ 27,
 29, 32
 font-style プロパティ 29,
 32
 font-weight プロパティ
 32

font 要素 27
 for 162
 シェル 200
 FOREIGN KEY 104
 FORMAT 93
 FORM 認証 125, 126
 form 要素 39
 FULLTEXT 105

G

GB2312 176
 GET 42, 43
 getAttribute 55
 getter 149
 GNU Emacs *see* Emacs
 GNU/Linux 197
 Google Web Toolkit 7
 GPL 107
 GREATEST 92
 GregorianCalendar 157
 GROUP BY 71

H

h1-h6 要素 17
 HashMap 163
 HashSet 162
 HAVING 71
 head 要素 24, 172
 height プロパティ 32
 HEX 91
 Hibernate 141
 href 属性 175
 HTML 9, 16
 テンプレート 24
 文字コード 172
 HTML Tidy 180
 HTML Validator 22, 27
 Html Validator 180
 html 要素 24
 HTTP 42
 httpd.conf 172
 HTTP ヘッダ 180

I

i18n *see* 国際化

IANA 172
 iBATIS 141
 IBM 特殊文字 169
 ICU 159
 IDE 8
 id 属性 29
 IE Tab 180
 IF 91
 シェル 200, 201, 203
 ifconfig 34, 199
 img 要素 20, 43, 138
 import 属性 47, 114
 IN 97
 INDEX 105
 InnoDB 81, 108
 input 要素 40
 INSERT 63, 99
 INSERT IGNORE 65
 INT 62
 INTEGER 62
 International Components
 for Unicode *see*
 ICU
 internationalization *see*
 国際化
 Internet Explorer 16,
 180
 ipconfig 34, 199
 IP アドレス 34, 199
 IS NOT NULL 77
 IS NULL 77
 ISO-2022 167
 ISO-2022-JP 167
 ISO-8859-1 74, 167, 176
 ISO/IEC 10646 168

J

J2EE 10
 J2SE Development Kit
 see JDK
 J2SE Runtime
 Environment *see*
 JRE
 JakartaCommons
 fileUpload 52

- Java 181
 文字コード 169, 172
 Java 2 Enterprise Edition
 see J2EE
 Java DataBase
 Connectivity *see*
 JDBC
 Java DB 107
 Java Persistence API
 see JPA
 Java Studio Creator 141
 java.util.JapaneseImperial
 Calendar 158
 JAVA_HOME 183
 JavaBeans 128
 javac (文字コード) 172
 JavaScript 2, 7, 51, 150
 JavaServer Pages *see*
 JSP
 javax.script 7
 JDBC 10, 109, 187
 文字コード 173
 JDBCRealm 125
 JDK 181
 インストール
 (GNU/Linux) 201
 インストール (Mac OS X)
 208
 インストール (Windows)
 181
 JIS X 0201 167
 JIS X 0208 168
 JIS X 0212 168
 JIS X 0213 168
 JIS X 0221 168
 JIS 参照文字 171
 join コマンド 84
 JPA 141
 JRE 181
 JSF 141
 JSP 46, 129
 文字コード 172, 176
 JSP Standard Tag Library
 see JSTL
 JSP シンタックス 46
- JSTL 53
 文字コード 176
 JUnit 141
- K**
- Knoppix 197
 Konqueror 199
- L**
- l10n *see* 地域化
 label 要素 40
 LAMP 10
 language 属性 47
 Latin 1 74, 167
 LEAST 92
 LEFT 91
 LF 83
 LGPL 107
 LIMIT 121
 LinkedList 160
 Linux *see* GNU/Linux
 List 162
 list-style-image プロパティ
 32
 list-style-type プロパティ
 18, 32
 Live CD *see* ライブ CD
 Live HTTP Headers 44,
 138, 145, 180
 li 要素 18
 LN 92
 LOAD 82
 local *see* ロケール
 localization *see* 地域化
 LOCATE 91
 LOG 92
 login-config 要素 125
 LONG 62
 LONGBLOB 62
 LONGTEXT 62
 Ludia 105
- M**
- m17n *see* 多言語化
 Mac OS X 108, 207
- Map 164
 margin 31
 margin プロパティ 32
 MAX 93
 MD5 93
 Meadow 179
 MEDIUMBLOB 62
 MEDIUMTEXT 62
 MemoryRealm 125
 mata 要素 172
 method 属性 39, 43
 Microsoft Excel 84
 MIN 93
 mkdir 200
 mkfs 199
 MOD 67, 92
 mod_proxy_ajp 125
 Model 128
 model 130
 module (Slax) *see* モ
 ジュール (Slax)
 Mono 11
 Monolithic JSP 128
 mount 199, 200
 MS932 4, 169
 multilingualization *see*
 多言語化
 MVC 8, 11
 MyISAM 81, 100, 108
 MySQL 107
 パスワード 185
 文字コード 173
 mysql 184
 MySQL Certifications
 12
 MySQL Connector/J
 109, 187
 mysqldump 105
- N**
- native2ascii 144
 NEC 特殊文字 169
 NetBeans 8
 Network File System
 see NFS

NFS 199
 NOT NULL 61
 NOT NULL 制約 104
 NTFS Utils 182
 NULL 61
 NUMBER 62

O

O/R マッピング 141
 ODBC 109
 ol 要素 18
 OpenOffice.org Calc 84
 OpenType 168
 Opera 16
 OperaView 181
 OPTIMIZE TABLE 101
 Oracle 107
 ORACLE MASTER 12
 ORDER BY 72
 out 50

P

padding 31
 padding プロパティ 32
 page 133
 pageContext 55
 pageEncoding 属性 47
 page ディレクティブ 47, 150
 PATH 182
 Pattern 154
 Perl 11
 PHP 10, 11, 108, 178
 phpMyAdmin 69, 75, 178
 PI 92
 Plain Old Java Object *see* POJO
 POJO 130
 position プロパティ 32
 POST 43
 PostgreSQL 105, 107
 POWER 92
 PreparedStatement 115

pre 要素 21
 PRIMARY KEY 62
 Properties Editor 188, 194
 Prototype 123
 Proxy Server 194
 p 要素 17

Q

QR Code 181
 q 要素 21

R

RADIANS 92
 Rails *see* Ruby on Rails
 RAND 92
 RDB 58
 RDBMS 58
 REAL 62
 Realm 125
 REFERENCES 104
 Relational Database Management System *see* RDBMS
 REPEAT 91
 REPLACE 91
 request 133
 RESET QUERY CACHE 99
 REVOKE 69
 RFC2718 175
 RFC3986 175
 RGB 29
 ROUND 92
 rows 112
 rowSpan 属性 40
 Ruby on Rails 10, 141

S

Safari 16
 sc 187
 scote 133
 script 要素 43, 138
 sed 200
 SELECT 65, 112

Selenium 141
 semantic web *see* セマンティック・ウェブ
 Senna 105
 Server VM 51, 181
 Servlet 128, 132
 文字コード 176
 session 133, 137
 session-timeout 要素 137
 Set 163
 SET NAMES 75
 setAttribute 55
 setMaxRows 121
 setter 149
 SGML Parser 180
 SHA1 93
 Shift_JIS 4, 167-169
 SHOW CREATE 75, 100
 SHOW DATABASES 60
 SHOW TABLES 61
 SHOW WARNING 81
 SIGN 92
 SIN 92
 SJIS 169
 sjis 169
 Slax 197, 198
 SOURCE 75, 105
 span 要素 27, 30
 SQL 7, 106
 SQL Server 107
 SQL-92 62
 sql:query 112
 sql:update 112
 SQLite 107
 SQL インジェクション 9, 114
 SQRT 92
 src 属性 20, 43, 138
 SSL 通信 126
 Statement 121
 STDDEV 93
 STDDEV_POP 93
 STDDEV_SAMP 93

stdout.log 50
 stored procedure *see* ス
 トアド・プロシジャ
 String 153
 StringBuffer 153
 strong 要素 21
 Struts 141
 style 属性 27
 SUBSTR 67, 91
 SUBSTRING_INDEX
 91
 SUM 93
 Sun 認定 Web コンポーネン
 トディベロッパ 11
 SYSIBM.SYSDUMMY1
 67
T
 table 要素 19, 31
 taglib ディレクティブ
 54, 55, 144, 176
 TAN 92
 TCP/IP モニター 44
 td 要素 19
 TeraTerm 75
 TEXT 62
 text-align プロパティ
 29, 32
 text-decoration プロパティ
 32
 text/css 28
 this 149
 throw 152
 th 要素 19
 TIMEDIFF 92
 TIMESTAMP 62
 timeStyle 属性 145
 title 要素 23
 Tomcat 9, 172
 インストール
 (GNU/Linux) 202
 インストール (Mac OS X)
 210
 インストール (Windows)
 187

サービス 188
 ログ 50
 toString メソッド 50
 TreeMap 164
 TreeSet 164
 TRIM 91
 TRUNCATE 68, 82, 92
 try 152
 tr 要素 19
 type 属性 40, 145

U

UCS 167, 168
 UCS Transformation
 Format *see* UTF
 ul 要素 18
 UML クラス図 147, 151
 UNHEX 91
 Unicode 167, 168, 170,
 171
 Unicode スカラー値 168
 Unihan Database Search
 Page 170
 UNION 98
 UNION ALL 98
 Universal multi-octet coded
 Character Set *see*
 UCS
 Unix 84
 UPDATE 67, 112
 URI 175
 URL 42
 url-patterl 要素 133
 USE 60
 useBodyEncodingForURI
 属性 176
 UTF-16 24, 167
 UTF-8 9, 24, 167, 171,
 179
 UTF-8N 83, 179
V
 value 属性 40
 VAR_POP 93
 VAR_SAMP 93

VARCHAR 62
 VARCHAR2 62
 View 129
 view 134
 View Source Chart 181
 visibility *see* 可視性
 visibility プロパティ 32
 VLOOKUP 84
 VMware Server 198

W

W3C 35
 Web *see* ウェブ
 Web Developer 180
 Web Server *see* ウェブ
 サーバー
 Web Tools Platform *see*
 WTP
 web-app 要素 126
 web.xml 126, 133, 172
 Web コンポーネントディベ
 ロッパ 11
 WHERE 65
 width プロパティ 29, 32
 Wikipedia 175
 Windows Vista 178
 Windows XP 178
 Windows-31J 4, 9, 75,
 167–169, 173
 windows-31j 169
 WORA 2
 World Wide Web *see*
 ウェブ
 Write Once, Run Anywhere
 see WORA
 WTP 8, 188
 日本語化 189
 WWW *see* ウェブ
X
 XAMPP
 アンインストール
 (Windows) 187
 インストール
 (GNU/Linux) 202

インストール (Mac OS X) 208
 インストール (Windows) 183
 パスワード 185
 XHTML 23
 テンプレート 24
 XHTML 1.0 Strict 23, 195
 XHTML 1.0 Transitional 195
 xml-styleSheet 28
 XML シンタックス 46
 XML 宣言 24
 XML データベース 106
 XSS *see* くるすさいと
 xyzyzy 179

Z

z-index プロパティ 32
 Zero Width No-Break Space *see* ZWNBS
 zsh 208
 ZWNBS 179

あ

アクセシビリティ 35
 アクセス管理 58
 アクセス権限 68
 アクセッサ 149
 アップロード 52
 アノテーション 151
 アプリケーションサーバー 1
 アルゴリズム 164
 暗黙オブジェクト 50
 異体字 169
 インターネット 3
 インデックス 99, 105
 削除 100
 作成 100
 インポート 77
 引用 21
 引用符 20, 25
 シェル 46

インライン要素 30
 ウェブ 3
 ウェブアプリ 1
 ウェブアプリケーションサーバー 9
 ウェブサービス 3
 ウェブサーバー 1
 文字コード 172
 ウェブデザイン 35
 ウェブ標準 35
 ウェブブラウザ 16
 文字コード 174
 ウェブページ 1
 エクスポート 78
 エスケープ 174
 エッジ 93
 エディター *see* テキスト
 エディター
 円記号 *see* ¥
 オブジェクト 147
 オブジェクト指向データベース 106
 オブジェクト指向プログラミング言語 7
 オープンソース・ソフトウェア 11

か

改行 20
 改行コード 83, 179, 208
 楷書体 170
 外部キー 81
 外部キー制約 104
 拡張機能 (Firefox) 180
 拡張子 178
 可視性 147
 箇条書き 18
 画像 20
 カーディナリティ 79
 環境変数 182
 システム— 183
 管理者 178
 キーボードリピート 198
 基本情報処理技術者 11
 キャッシュ 99

行
 検索 65
 更新 67
 削除 68
 挿入 63
 教育漢字 168
 強調 21
 共有フォルダ 199
 クチ高 170
 クッキー *see* cookie
 クラス 29, 147
 クラスライブラリ 153
 グラフ 93
 グレゴリオ暦 157
 クロスサイト・スプリブティ
 ング 9, 54
 継承 151
 権限 68
 検索 65
 康熙字典 170
 更新 67
 構造体 (C 言語) 147, 150
 互換モード 24
 国際化 142
 国際文字集合 *see* UCS
 小塚明朝 Pro-VI 168
 コネクション・プール 116
 コマンドプロンプト 179
 文字コード 173
 コミット 82
 コメント (SQL) 61
 コレクション 159
 コンストラクタ 150
 コンソール 50
 コントローラ *see* Controller
さ
 削除
 行 68
 データベース 60
 テーブル 61
 作成

テーブル 60
 サービス (Tomcat) 188
 サブクエリ 97
 サブレット *see* Servlet
 シェルスクリプト 46
 式言語 *see* EL
 式タグ 50
 字形 169
 システム環境変数 183
 字体 169
 実体関連図 *see* ER 図
 シート 58
 終了タグ 17, 20, 25
 重力加速度 170
 主キー制約 62, 104
 情報処理技術者試験 11
 常用漢字 168
 書体 170
 シンボリック・リンク
 182
 人名用漢字 168
 数値文字参照 25
 スキーマ 58
 スクリプトレット 46
 スコープ 133
 スタイルシート 26, 180
 スタイル・ファイル 28
 スタアド・プロシジャ
 101
 スーパークラス 151
 正規化 8, 79
 正規表現
 Java 153
 MySQL 66
 整形済みテキスト 21
 制約 61, 104
 セキュリティ
 MySQL 185
 XAMPP 185
 セキュリティ・ホール 8
 セッション *see* session
 セット 162
 セマンティック・ウェブ
 26
 セルの連結 40

セレクタ 29
 宣言タグ 50
 センタリング 29
 挿入 (行) 63
 属性 20, 25, 147
 ソフトウェア開発技術者
 12
た
 第1~4水準漢字 168
 高島屋 169
 タグ 17
 終了— 17
 多言語化 142
 妥当な HTML 文書 25
 ターミナル
 GNU/Linux 13
 Mac OS X 38
 文字コード 173
 段落 17
 地域化 142
 通信ヘッダ 180
 ツチ吉 169
 ディスク
 フォーマット 199
 マウント 200
 ディスクイメージ 198
 ディレクトリの作成 200
 データ構造 159
 テキストエディター
 GNU/Linux 200
 Mac OS X 207
 Windows 179
 テクニカルエンジニア 11
 デザインパターン 141
 テスト 141
 データ型
 RDBMS 62
 データの独立性 57
 データベース 2, 59
 削除 60
 利用 60
 データベース管理システム
 see DBMS
 データモデリング 79

デフォルト・コンストラクタ
 150
 テーブル 58
 削除 61
 作成 60
 デプロイメント記述子
 see web.xml
 テンプレート (XHTML)
 24
 統合開発環境 8
 特殊文字 25
 トランザクション 81, 82
な
 並び替え 164
 二重引用符 25
 認証 125
 ノード 93
は
 排他制御 57
 配列 160
 ハシゴ高 169
 パスワード
 MySQL 185
 XAMPP 185
 パーセントエンコーディング
 174
 バックスラッシュ *see* \
 パーティションの作成
 199
 パディング *see* padding
 ヒアドキュメント (シェル)
 204
 非推奨要素 27
 ビット演算 93
 秀丸 179
 ビュー *see* View
 表
 HTML 19
 RDB 58
 表計算ソフト 58
 表計算ソフト 58
 標準タグライブラリ *see*
 JSTL

- 標準偏差 93
 標準モード 24
 標準ユーザ 178
 ファイルシステムの作成
 199, 200
 フォーマット(ディスク)
 199
 フォーム 38
 文字コード 174
 フォントサイズ 27
 副問い合わせ 97
 符号化方式 4, 167
 符号化文字集合 167
 プランナ 99
 フルテキスト・インデックス
 100, 105
 フレーム 31
 フレームワーク 141
 プロキシ・サーバー 179,
 194
 プロジェクト(Eclipse)
 47
 ブロックレベル要素 30
 プロトコル 42
 プロパティ 29
 プロパティエディター
 see Properties Editor
 プロパティ・ファイル
 143
 分散 93
 文書型宣言 24
- ページ・レイアウト 31
 ヘッダ(HTTP) 180
 変数 147
 Java 147
 SQL 73
 包摂 169
 ボーダー *see* border
 ボックス 31
 ボックスモデル 31
 ポート 42
- ま**
- マウント 200
 マークアップ言語 17
 マージン *see* margin
 マップ 163
 見出し 17
 明朝体 170
 メソッド 147
 メーラー 2
 文字コード 4, 167
 HTML 24
 Java 169
 MySQL 74
 Wikipedia 175
 メール 167
 文字参照 25
 文字実体参照 25
 文字集合 167
 文字化け 9
 モジュール(Slax) 200
 モダン・ブラウザ 16
- モデル *see* Model
- ゆ**
- ユーザ認証 125
 ユニーク制約 104
 ユニバーサルデザイン 35
 ユリウス暦 157
- ら**
- ライブCD 13, 197
 リクエスト 9
 リスト 160
 リバースポイント 182
 リファラー 138
 リレーショナル・データベー
 ス *see* RDB
 リレーショナル・データベー
 ス管理システム *see*
 RDBMS
 リンク 43
 ループ 162
 レイアウト 31
 例外 151
 連想配列 163
 ログ(Tomcat) 50
 ロケール 142
 ロール 125
- わ**
- ワークスペース(Eclipse)
 189