

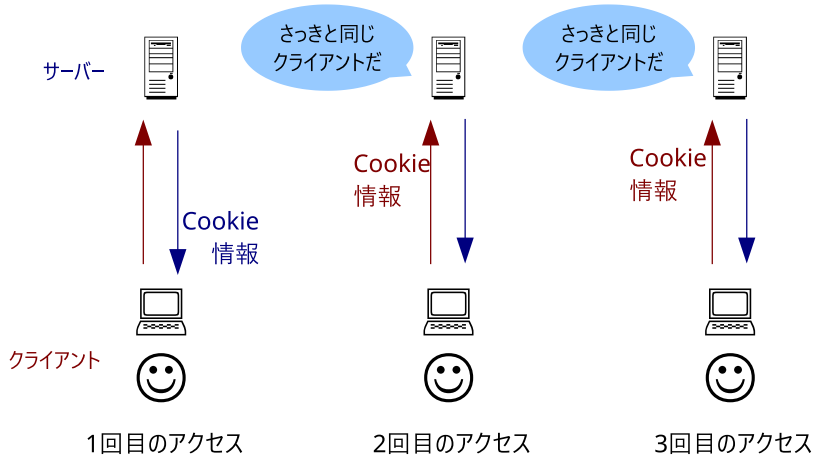
第III章 セッションの利用

III.1 セッションとは

Web サーバーと Web ブラウザー間の通信 (HTTP による通信) は本来一回ページを表示するごとに完結するものである。つまり、ユーザーが過去にどのようなページを見ていたか、などといった履歴には関係なく動作する。

そこで、過去の履歴 (例えば会員制ページのユーザーのログイン情報や、ショッピングサイトの商品の選択 (いわゆるショッピングカート) の情報) に依存するような Web アプリケーションを実現するためには、なんらかの形で過去のユーザーのアクセスの情報を保持する必要がある。このようなデータは、Web サイトにアクセスするユーザーごとに管理しなければならないので、単純にフィールドやファイルなどに保存することはできない。このためサーバー側にユーザーごとにデータを保持し、ブラウザからのページ要求のたびに、何らかの方法でユーザーを識別するためのデータを送信してもらう方法を取る。

このユーザーを識別するデータを送信する方法としては、クッキー (cookie) という技術を使う方法、フォームの隠し要素 (hidden) を使う方法、URL 中の Query String に履歴データを埋め込む方法などがある。いずれにしても、店舗の“会員証”・病院の“診察券”に相当するものをブラウザに渡して、来店・来院する (つまり、ページにアクセスする) たびに提示してもらうようなものである。



問 III.1.1 クッキー (cookie) とは、どういう仕組みか、調べよ。

Java Servletではユーザーごとのデータを扱うためにセッション (session) (もともとは会期などという意味) という仕組みを提供している。セッションは裏側ではクッキーなどの仕組みを使っているが、複雑な部分をプログラマーが見なくても済むようにして、Servletのプログラマーが簡単にユーザーの履歴データを利用できるインタフェースを提供している。使い方は簡単で、HttpServletRequestクラスのgetSessionというメソッドでセッションオ

ブジェクト（店舗の“顧客情報”・医院の“カルテ”に相当する）を取り出し、そのオブジェクトに対して、データを関連づけ（setAttribute）たり、読み出し（getAttribute）たりするだけである。クッキーの発行や確認は Web アプリケーションサーバーが舞台裏で行っているため、プログラマーが行う必要はない。

セッションはユーザーごとに用意されるので、一連のアクセスで以前にセットした値を利用することができる。一方、他のユーザーがセットしたデータは見えない。

III.2 セッションを利用したアクセスカウンター

まず、セッションを利用したアクセスカウンターを紹介する。一見、ファイルを利用したアクセスカウンターとそれほど違いはないが、ユーザーごとにカウンターのデータを保持する。これは異なる Internet Explorer や Firefox など別のブラウザでアクセスすると（Servlet からは別のユーザーに見えるので）振舞いの違いを確認することができる。

ファイル `SessionCounter.java`

```
1 import java.io.IOException;
2 import java.io.PrintWriter;
3
4 import jakarta.servlet.ServletException;
5 import jakarta.servlet.annotation.WebServlet;
6 import jakarta.servlet.http.HttpServlet;
7 import jakarta.servlet.http.HttpServletRequest;
8 import jakarta.servlet.http.HttpServletResponse;
9 import jakarta.servlet.http.HttpSession;
10
11 @WebServlet("/SessionCounter")
12 public class SessionCounter extends HttpServlet {
13     private static final String COUNTER = "counter";
14
15     @Override
16     protected void doGet(HttpServletRequest request,
17                           HttpServletResponse response)
18         throws ServletException, IOException {
19         response.setContentType(
20             "text/html; charset=UTF-8");
21         HttpSession session = request.getSession(true);
22         int i = 0;
23         try {
24             i = (int) session.getAttribute(COUNTER);
25         } catch (NullPointerException
26                | NumberFormatException e) {
27             /* i = 0 のまま */
28         }
29         PrintWriter out = response.getWriter();
30         out.println("<html><head></head><body>");
31         out.printf("あなたの %d 回目の御訪問です。", i++);
32         out.println("</body></html>");
33         session.setAttribute(COUNTER, i);
34         out.close(); // closeを忘れない
35     }
```

HttpServletRequest クラスの getSession メソッドで現在のセッションオブジェクト (HttpSession クラスのオブジェクト) を得る。引数の true はセッションオブジェクトがなければ作成することを示す。

Session クラスの getAttribute メソッドでその値を取り出す。

getAttribute メソッドの引数は取り出したいデータに対応づけられた名前前で、戻り値がセッションに保存されていたその名前のデータである。

getAttribute メソッドの戻り値型は Object 型 (つまりすべてのクラスのスーパークラス) なので、String 型や int 型などに型変換 (ダウンキャスト・ナローイング) する必要がある。Object 型から int への型変換は int 型のラッパークラスの Integer 型を経由して行なわれる。

最初のアクセスではセッションにデータが保存されていないので、例外が発生し、i の値は 0 になる。

最後に setAttribute メソッドを用いて、セッションオブジェクトにデータを保存する。setAttribute メソッドの第1引数は String 型であり、保存するデータに対応させる名前である。この名前は後で getAttribute でデータを取り出すときに用いる。setAttribute メソッドの第2引数は実際に保存するデータである。ここにはどのような型のデータが来ることもあり得るため、setAttribute の第2引数の型は、すべてのオブジェクトの型を包含する Object という型になっている。Object はすべてのクラスのスーパークラスである。

int 型から Integer 型への型変換 (オートボックス) と Integer 型から Object 型への型変換 (アップキャスト・ワイドニング) は暗黙的に行なわれる。

問 III.2.1 アクセス時の時刻をセッションに保存し、次のアクセス時に「前回は何月何日何時何分何秒にアクセスしました」 (または「初めてか久しぶりのアクセスです」) と表示するサーブレット AccessTime.java を (getLastAccessTime メソッドを使わずに) 作成せよ。

問 III.2.2 アクセス時に、乱数 (`Math.random()`) で 6 ~ 10 の整数を生成し、「今回の来店ポイントは?点です。累計のポイントは?点になりました。」と (?の部分置き換えて) 表示するサーブレットを作成せよ。

なお、ポイントをファイルに保存する必要はない。つまり、サーバーをシャットダウンしたり、時間が経ってセッションがタイムアウトしたときは 0 に戻っても構わない。

セッションとしてちゃんと動作していること (Chrome と Firefox など 2 種類のブラウザから同時にアクセスして独立に動くこと) を確認しておくこと。

III.3 Quiz サーブレット

セッションを利用するもう少し大きな Servlet として Quiz サブレットを例にあげる。これは過去に正解した問題数などによって、表示を変更する Servlet である。

正解した問題数や現在何問めを実行しているかを保存するためにセッションを利用する。（さもないと、複数のユーザーが同時にこのサブレットにアクセスしたときに、正解した問題数のデータがごっちゃになってしまう。）

例題: QUIZ

ファイル `quiz.txt`

```
1 日本で一番高い山は？ エベレスト 富士山 飯野山 2
2 日本で一番広い湖は？ 琵琶湖 府中湖 満濃池 浜名湖 1
3 1998年のワールドカップ開催地は？ 日本 フランス ブラジル 2
4 讃岐三白でないのは？ 綿 塩 砂糖 うどん 4
5 香川大学創造工学部の所在地は？ 幸町 花園町 林町 多肥上町 3
6 香川県の県庁所在地は？ 徳島 松山 坂出 尾道 高松 津名 5
```

この Servlet は上のようなテキストファイルから右のような QUIZ を表示するページを作成する Servlet である。

ようこそ QUIZへ!
では最初の問題です。

問: 日本で一番高い山は?

エベレスト 富士山 飯野山

この Servlet では“現在何番目の問題か?” (number)と“これまでの正解数” (score) というデータがセッションのなかに保持されている。(その1)の部分は特に変わったところはない。ArrayList を使用するのでこれを import している。また、いくつかの定数を定義している。

ファイル `Quiz.java (その1)`

```
1 import java.io.BufferedReader;
2 import java.io.File;
3 import java.io.FileInputStream;
4 import java.io.IOException;
5 import java.io.InputStreamReader;
6 import java.io.PrintWriter;
7 import java.util.ArrayList;
8
9 import jakarta.servlet.ServletException;
10 import jakarta.servlet.annotation.WebServlet;
11 import jakarta.servlet.http.HttpServlet;
12 import jakarta.servlet.http.HttpServletRequest;
13 import jakarta.servlet.http.HttpServletResponse;
14 import jakarta.servlet.http.HttpSession;
15
16 @WebServlet("/Quiz")
17 public class Quiz extends HttpServlet {
18     private static final String ANSWER = "answer";
19     private static final String SCORE = "score";
20     private static final String NUMBER = "number";
```

```
21 private static final String QUESTIONS = "questions";
```

(その2)のdoGetメソッドは最初にServletが実行されるときに呼ばれる。このメソッドは単にdoPostメソッドを呼び出すだけである。

ファイル Quiz.java (その2)

```
23 @Override
24 protected void doGet(HttpServletRequest request,
25                       HttpServletResponse response)
26     throws ServletException, IOException {
27     // 最初の問は GET
28     doPost(request, response);
29 }
```

(その3)はdoPostメソッドの最初の部分を示す。ここでは、いくつかの局所変数を宣言していて、セッションオブジェクトを作成している。

ファイル Quiz.java (その3)

```
31 @Override
32 protected void doPost(HttpServletRequest request,
33                       HttpServletResponse response)
34     throws ServletException, IOException {
35     response.setContentType(
36         "text/html; charset=UTF-8");
37     PrintWriter out = response.getWriter();
38     out.println("<html><head></head><body>");
39
40     int i, number = 0, score = 0;
41     ArrayList<String[]> questions;
42
43     HttpSession session = request.getSession(true);
```

session.isNew()が真のときは、セッションができたばかりであることを示す。つまり、このページのアクセスが最初の問であることがわかる。

そのときは、QUIZのデータが入っているファイル(quiz.txt)を開いて、questionsというArrayListに読み込む。このquestionsの値を次の問以降の表示のときに利用するために、setAttributeメソッドを用いて、セッションオブジェクトに保存する。この例では、ArrayList<String[]>やStringなどの型からObject型への型変換が起こっているが、このような上向きの型変換は暗黙に行なわれる。

また、最初の問用のメッセージを表示する。

ファイル Quiz.java (その4)

```
45     if (session.isNew()
46         || session.getAttribute(QUESTIONS) == null) {
47         // 最初の問
48         questions = new ArrayList<String[]>();
49         File f = new File(getServletContext()
50             .getRealPath("/WEB-INF/quiz.txt"));
51         BufferedReader in
52             = new BufferedReader(new InputStreamReader(
53                 new FileInputStream(f), "UTF-8"));
54         String line="";
```

```

55     while ((line = in.readLine()) != null) {
56         line = line.trim();
57         if (line.trim().equals("")) continue;
58         // 空白の1つ以上の繰返し
59         questions.add(line.split("\\s+"));
60     }
61     in.close();
62     session.setAttribute(QUESTIONS, questions);
63     out.println("<p>ようこそ QUIZへ!<br />");
64     out.println("     では最初の問題です。</p>");
65 }

```

一方、`session.isNew()` が偽のときは、最初の間ではないので、送られたフォームのデータから、前の問題で解答者が選んだ答の番号 (`answer`) を得る。また、セッションデータには最初の間のように読み込んだ問題のデータ、現在の問題の番号 (`"number"`) とこれまでの正解数 (`"score"`) が記録されているはずなので、`HttpSession` クラスの `getAttribute` メソッドでその値を取り出す。

`questions` の `number - 1` 番目の最後のトークンを解答と比べる。その結果によってメッセージと `score` 変数の値を変える。

ファイル `Quiz.java` (その 5)

```

66     else {
67         // 最初の間ではない
68         try {
69             number = (int)session.getAttribute(NUMBER);
70             score = (int)session.getAttribute(SCORE);
71             questions = (ArrayList<String[]>)
72                 session.getAttribute(QUESTIONS);
73
74
75             String[] tokens = questions.get(number - 1);
76             int a = Integer.parseInt(
77                 tokens[tokens.length - 1]);
78             int answer = Integer.parseInt(
79                 request.getParameter(ANSWER));
80             if (a == answer) { // aは最後の文字
81                 out.println("正解です。<br />");
82                 score++;
83             } else {
84                 out.println("残念でした。<br />");
85             }
86         } catch (Exception e) {
87             session.removeAttribute(QUESTIONS);
88             out.println(
89                 "想定外のアクセスでエラーが起きました。"
90                 + "タブを閉じるかリロードしてください。");
91             e.printStackTrace(out);
92             out.println("</body></html>");
93             out.close();
94             return;
95         }
96     }

```

次に、次の問を表示する準備をする。ただし、`number` 番めの問題がないときは、クイズを終了する。

ファイル Quiz.java (その 6)

```
98     if (number >= questions.size()) { // 終
99         out.println("<br />");
100        out.println("これで QUIZは終わりです。<br />");
101        out.printf("正解数は、%d問でした。%n", score);
102        session.removeAttribute(QUESTIONS);
103    }
```

問題が終わりでなければ、questions から次の問の情報を読み取って、フォームとして出力する。そして各選択肢のラジオボタンと送信ボタン、リセットボタンを出力する。form タグに action 属性がないが、その場合は自分自身（現在表示中のページと同じ URL）にフォームデータを送信する。

最後に setAttribute メソッドを用いて、セッションオブジェクトにこれまでの問題数と正解数を記録している。number を一つ増分してから、setAttribute を呼び出して、number と score をセッションオブジェクトに書き込んでいる。（このデータは次の問題を表示するときに利用される。）

ファイル Quiz.java (その 7)

```
105     String[] tokens = questions.get(number);
106     out.println("次の問: " + tokens[0] + "<br />");
107     out.println("<form method='post'>");
108     for (i = 0; i < tokens.length - 2; i++) {
109         out.print("<input type='radio'");
110         out.print(" name='answer'");
111         out.printf(" value='%d' /> %s",
112             i + 1, tokens[i + 1]);
113     }
114     out.println("<br />");
115     out.print("<input type='submit'");
116     out.println(" value='送信' />");
117     out.print("<input type='reset'");
118     out.println(" value='やめ' />");
119     out.println("</form>");
120
121     number++;
122     session.setAttribute(NUMBER, number);
123     session.setAttribute(SCORE, score);
124 }
125 out.println("</body></html>");
126 out.close();
127 }
128 }
```

なお、この Servlet を、リロードボタンや戻るボタンなどがクリックされた場合、複数のタブで同時に開いた場合などに、適切に対応するように改良することはなかなか難しい。適切に対応することが必要なときは、生の Servlet ではなく、Wicket などの Web アプリケーションフレームワークを採用することを検討するほうが良いだろう。

問 III.3.1（選択肢の記録） Quiz.java を、正解数だけではなくて、各問の選んだ選択肢、正誤までわかるように記録し、その結果を各問の問題文、正答ともに「これで QUIZ は終わりです。」のメッセージのあとにテーブルに整形して表示するサーブレット QuizEx.java を作成せよ。QUIZ の問題数は何問に

なるかわからないので、問題数に依存しないようにすること。選択肢・正答は番号だけの表示は不可である。（下の表示例を参考にすること。）

残念でした。

これで QUIZ は終了です。

番号	問題	正解	解答	正誤
第1問	日本で一番高い山は?	富士山	飯野山	×
第2問	日本で一番広い湖は?	琵琶湖	府中湖	×
第3問	1998年のワールドカップ開催地は?	フランス	日本	×
第4問	讃岐三白でないのは?	うどん	うどん	○
第5問	香川大学創造工学部の所在地は?	林町	林町	○
第6問	香川県の県庁所在地は?	高松	尾道	×

正解数は、6問中2問でした。

問 III.3.2 (オプションの選択) 次のようなオプションとその価格が書かれたテキストファイル

ファイル `pasocon.txt`

```
1 筐体 タワー 20000 スリム 35000 ミニ 50000
2 CPU Celeron430 4000 DuoE8400 20000 QuadQ9450 37000
  QuadQ9550 62000
3 RAM 512MB 2000 1GB 4000 2GB 6000 4GB 9000
4 HDD 80GB 4000 250GB 6000 320GB 6500 500GB 8000 1TB
  20000
5 光学D DVD-R/RW 5000 Blu-ray 20000
```

から、次のようなオプション構成を選択できるページを各パーツ毎に作成し、

```
1 <html>
2 <body>
3 CPUを選んで下さい
4 <br />
5 <form method='post'>
6 <input type='radio' name='answer' value='1' />
7 Celeron430 4000円
8 <input type='radio' name='answer' value='2' />
9 DuoE8400 20000円
10 <input type='radio' name='answer' value='3' />
11 QuadQ9450 37000円
12 <input type='radio' name='answer' value='4' />
13 QuadQ9550 62000円
14 <br />
15 <input type='submit' value='送る' />
16 <input type='reset' value='キャンセル' />
17 </form>
18 </body>
```


すべてのパーツの選択肢を選んだ時点で、選択した構成の合計価格を表示するサーブレット `SelectOptions.java` を作成せよ。さらに、見積書のように表の形に整形せよ。

キーワード:

クッキー, `hidden` (隠し要素), セッション, `HttpSession` クラス,
`getSession` メソッド, `getAttribute` メソッド, `setAttribute` メソッド,
`isNew` メソッド, `invalidate` メソッド,

