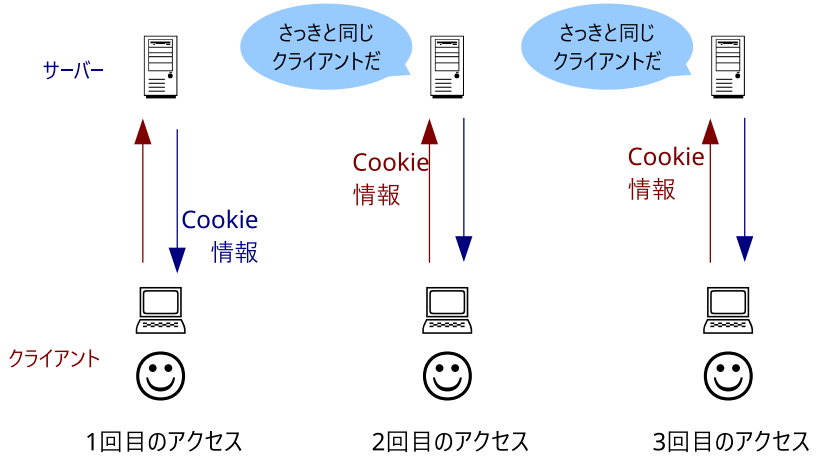
第III章 セッションの利用

III.1 セッションとは

Web サーバーと Web ブラウザーの間の通信 (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、ブラウザー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”・医院の“診察券” に相当するものをブラウザーに渡して、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 アプケーションサーバーが舞台裏で行っているので、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 など別のブラウザーでアクセスすると（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.printf("""
31         <!Doctype html>
32         <html><head></head><body>
33         あなたの %d 回目の御訪問です。
34         </body></html>
35         """, i++);
36     session.setAttribute(COUNTER, i);
37     out.close();  // closeを忘れない
38   }
```

```
39 }
```

`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（オートボクシング）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、次回のアクセス時に「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（？の部分を置き換えて）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 サー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。

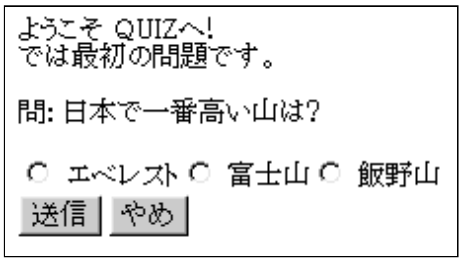
正解した問題数や現在何問めを実行しているかを保存するためにセッションを利用する。（さもないと、複数のユーザーが同時にこのサーブレットにアクセスしたときに、正解した問題数のデータがごっちゃになってしまう。）

例題: QUIZ

ファイル quiz.txt

```
日本で一番高い山は？　エベレスト　富士山　飯野山　2
日本で一番広い湖は？　琵琶湖　府中湖　満濃池　浜名湖　1
1998年のW杯開催地は？　日本　フランス　ブラジル　2
讃岐三白でないのは？　綿　塩　砂糖　うどん　4
香川大学創造工学部の所在地は？ 幸町　花園町　林町　伏石町　3
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（各項目はタブ文字で区切られている）から右のようなクイズを表示するページを作成する Servlet である。



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）

```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.ArrayList;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

@WebServlet("/Quiz")
public class Quiz extends HttpServlet {
  private static final String ANSWER  = "answer";
  private static final String SCORE   = "score";
  private static final String NUMBER  = "number";
```

```
21    private static final String QUESTIONS = "questions";
22
```

（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    }
30
```

（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）

```
70    @Override
71    protected void doPost(HttpServletRequest request,
72                          HttpServletResponse response)
73              throws ServletException, IOException {
74      response.setContentType(
75        "text/html; charset=UTF-8");
76      PrintWriter out = response.getWriter();
77      out.println("<!Doctype html>");
78      out.println("<html><head></head><body>");
79
80      int number = 0, score = 0;
81      ArrayList<String[]> questions;
82
83      HttpSession session = request.getSession(true);
84
```

次の部分では 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、クイズのデータを questions という ArrayList に読み込む。readQuizFile メソッドの定義は下に示す。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）

```
85      if (session.isNew()
86        || session.getAttribute(QUESTIONS) == null) {
87        // 最初の問
88        questions = readQuizFile();
89        session.setAttribute(QUESTIONS, questions);
```

```
90          out.println("""
91              <p>ようこそ QUIZへ!<br />
92              では最初の問題です。</p>""");
93      }
```

ここで readQuizFile は次のように定義されているメソッドである。クイズのデータが入っているファイル（quiz.txt）を開いて読み込む。

ファイル Quiz.java（その 5）

```
31  private ArrayList<String[]> readQuizFile()
32                                  throws IOException {
33      ArrayList<String[]> qs = new ArrayList<>();
34      File f = new File(getServletContext()
35                  .getRealPath("/WEB-INF/quiz.txt"));
36      try (BufferedReader in
37        = new BufferedReader(new InputStreamReader(
38            new FileInputStream(f), "UTF-8"))) {
39          String line="";
40          while ((line = in.readLine()) != null) {
41            line = line.trim();
42            if (line.trim().equals("")) continue;
43            // タブの１つ以上の繰返し
44            qs.add(line.split("\t+"));
45          }
46      }
47      return qs;
48  }
49
```

一方、session.isNew() が偽のときは、最初の問ではないので、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問のとき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 6）

```
94      else {
95        // 最初の問ではない
96        try {
97          number = (int)session.getAttribute(NUMBER);
98          score  = (int)session.getAttribute(SCORE);
99          questions = (ArrayList<String[]>)
100                       session.getAttribute(QUESTIONS);
101
102         String[] tokens = questions.get(number - 1);
103         int a = Integer.parseInt(
104                   tokens[tokens.length - 1]);
105         int answer = Integer.parseInt(
106                        request.getParameter(ANSWER));
107         if (a == answer) { // a は最後の文字
108           out.println("正解です。<br />");
109           score++;
110         } else {
```

```
111             out.println("残念でした。<br />");
112           }
113         } catch (Exception e) {
114           session.removeAttribute(QUESTIONS);
115           out.println("""
116               想定外のアクセスでエラーが起こりました。
117               タブを閉じるかリロードしてください。""");
118           e.printStackTrace(out);
119           out.println("</body></html>");
120           out.close();
121           return;
122         }
123       }
124
```

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 7）

```
125       if (number >= questions.size()) { // 終
126         out.printf("""
127             <br />
128             これで QUIZは終わりです。<br />
129             正解数は、%d問でした。
130             """, score);
131         session.removeAttribute(QUESTIONS);
132       }
```

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、outputQuiz メソッドで出力する。このメソッドの定義は下に示す。

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 8）

```
133       else { // 次の問を表示
134         String[] tokens = questions.get(number);
135         outputQuiz(out, tokens);
136         number++;
137         session.setAttribute(NUMBER, number);
138         session.setAttribute(SCORE, score);
139       }
140       out.println("</body></html>");
141       out.close();
142     }
143 }
```

ここで outputQuiz メソッドは、クイズを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、フォームは自分自身（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。

ファイル Quiz.java（その 9）

```
50   private void outputQuiz(PrintWriter out,
51                           String[] tokens) {
52       out.printf("次の問: %s <br />", tokens[0]);
53       out.println("<form method='post'>");
54       for (int i = 0; i < tokens.length - 2; i++) {
55         out.printf("""
56             <input type='radio'
57                    name='answer'
58                    value='%d' /> %s
59             """,  i + 1, tokens[i  + 1]);
60       }
61       out.println("""
62           <br />
63           <input type='submit'
64                  value='送信' />
65           <input type='reset'
66                  value='やめ' />
67           </form>""");
68   }
69
```

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。適切に対処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。クイズ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W杯開催地は?	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

```
筐体　タワー 20000　スリム 35000　ミニ　50000
CPU Celeron430　4000　DuoE8400　20000　QuadQ9450　37000　QuadQ9550　62000
RAM 512MB　2000　1GB 4000　2GB 6000　4GB 9000
HDD 80GB　4000　250GB　6000　320GB　6500　500GB　8000　1TB 20000
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、

```
<!Doctype html>
<html>
<body>
CPUを選んで下さい
<br />
<form method='post'>
<input type='radio' name='answer' value='1' />
Celeron430 4000円
<input type='radio' name='answer' value='2' />
DuoE8400 20000円
<input type='radio' name='answer' value='3' />
QuadQ9450 37000円
<input type='radio' name='answer' value='4' />
QuadQ9550 62000円
<br />
<input type='submit' value='送る' />
<input type='reset'  value='キャンセル' />
</form>
</body>
</html>
```

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,