情報科学プロジェクト実験 コンピュータシステム研究室
HTTP Cookie

HTTPでは1回ごとの要求がそれぞれ独立に扱われるように設計されている。
このため状況に応じて応答の内容を変えるような動的なコンテンツ生成を
考えた場合には、ある要求をしてきたクライアントが以前に要求をした
クライアントと同じであるかを確認する必要がある。一連の関係を持った
要求の管理をセッション管理という。(session: ある特定の活動を行う期間）

セッション管理を行う方法の一つにHTTP Cookieがある。これは、ある要求を
行ったクライアントに、他と区別できる唯一の値（文字列）を与えておき、
次回に要求を行う際にそのクライアントに与えた値を提示させる方法である。
（顔の似た複数の犬がいたときにある一匹の犬にクッキーを与えれば、
口の回りにクッキーをつけたその犬をしばらくの間だけ区別できる？）

日常で行われる似たセッション管理は銀行の窓口で行われている方法である。
窓口で送金や引き下ろしの手続きをすると、番号札が渡された上で椅子にすわって
待つように促される。数分後に名前が呼ばれるがその際に渡された番号札を提示する
ことでサービスを受けている本人であることが確認される。
この番号札に相当するのがCookieである。

HTTP CookieではWebクライアントにCookieを渡す方法とそれを提示する方法が
定められており、多くのWebブラウザがこれに対応している。手順は以下の通り。
1.あるHTTP要求に対してサーバは応答ヘッダにSet-Cookie:を設定する。
2.Webブラウザは設定されたその値をメモリまたはファイルに保存する。
3.Webブラウザは指定されたURLへのアクセスの際には、要求ヘッダにCookie:で
その値を指定する。
4.サーバは要求ヘッダ内のCookie:の指定の有無で応答内容を変更する。
(この値はCGIの環境変数HTTP_COOKIEに設定される）

補足：
HTTP CookieはRFC 2109, RFC 2965, RFC 6265で標準化されているが
この方法を考えた元々のネットスケープ社の方法と異なっており、
標準にも関わらず細かい部分で勝手に変更されて使用されている。

以下がその例である。
1.応答ヘッダにSet-Cookie:の設定

```
--------------------------------
$ telnet 192.168.114.15 80
Trying 192.168.114.15...
Connected to 192.168.114.15.
Escape character is '^]'.
GET /~okam/cgi-bin/coo.cgi HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 03 Dec 2014 11:58:16 GMT
Server: Apache/2.4.10 (Fedora)
Set-Cookie: name=abcdefg; expires=Fri, 13 Dec 2013 16:05:00 JST; Path=/~okam/cgi-bin/
Connection: close
Content-Type: text/plain; charset=UTF-8

cookie test
Connection closed by foreign host.
--------------------------------
```

この例では、
Set-Cookie: name=abcdefg; expires=Fri, 13 Dec 2013 16:05:00 JST; Path=/~okam/cgi
-bin/
の部分が応答ヘッダに指定されたCookieである。
次回以降の要求では name=abcdefg を提示することが求められている。
expiresは有効期限でありこの時刻までこのCookieが使用できるので、ブラウザは
ファイルに保存する。PathはCookieを提示する条件でこのディレクトリ以下にある
すべてのURLの要求に対して指定されたCookieが使える。

3,4:要求ヘッダでのCookie指定とその応答

```
--------------------------------
$ telnet 192.168.114.15 80
Trying 192.168.114.15...
Connected to 192.168.114.15.
Escape character is '^]'.
GET /~okam/cgi-bin/coo.cgi HTTP/1.0
Cookie: name=abcdefg

HTTP/1.1 200 OK
Date: Wed, 03 Dec 2014 11:59:42 GMT
Server: Apache/2.4.10 (Fedora)
Set-Cookie: name=abcdefg; expires=Fri, 13 Dec 2013 16:05:00 JST; Path=/~okam/cgi-bin/
Connection: close
Content-Type: text/plain; charset=UTF-8

cookie test
name=abcdefg
Connection closed by foreign host.
--------------------------------
```

Webクライアントからの要求として、Cookie: name=abcdefg を要求ヘッダに
追加した。その結果、応答にそのクッキーをそのまま表示している。

CGIは以下の通りである。

```
--------------------------------
#include <iostream>
#include <cstdlib>
#include <string>
using namespace std;

int main()
{
   // text/plainを指定していることに注意
   cout << "Content-type: text/plain; charset=UTF-8\r\n"
        << "Set-Cookie: name=abcdefg; " // abcdefgはランダムに選ぶ必要がある
            "expires=Sat, 11 Jan 2013 16:05:00 JST; "
            "Path=/~okam/cgi-bin/"
        << "\r\n\r\n";

   cout << "cookie test\n";
   if (const char * e = getenv("HTTP_COOKIE")) {
      string s = e;
      cout << s << endl;;
   }
   return 0;
}
--------------------------------
```

Cookieの指定方法：
NAME=VALUE をセミコロンまたはコンマで区切って指定する。
NAMEやVALUEには、セミコロン、コンマの含まない文字列を指定できる。
（なぜかVALUEの方をダブルクォート"で囲まない)
NAMEの部分は英数文字であるが英文字は大文字小文字の区別がない。
有効期限はNAMEをexpiresとし、以下の形式で指定する。
   expires=Sun, 10 Jun 2001 12:00:00 GMT
有効期限に過去を設定するとブラウザ上のCookieは直ちに削除される
pathを設定すると次回のURLと最長の前方一致したCookieが選ばれる
domainを指定する次回のURLのホストの後方一致でCookieが選ばれる
secureを指定するとhttps通信の時のみCookieが使われる

以下は有用と思われるプログラム

```
--------------------------------
// 指定時刻をcookieで利用できる形式の文字列で返す関数
#include <iostream>
#include <string>
#include <ctime>

std::string
get_cookie_time(time_t t)
{
   const int sz = 32; // 30 is enough for this purpose
   char a[sz];
   strftime(a, sz, "%a, %d %b %Y %T %Z", localtime(&t));
   return a;
}

int main()
{
  time_t t = time(0);  // 現在時刻の取得
  std::cout << get_cookie_time(t + 0) << "\n";
  std::cout << get_cookie_time(t + 3600) << "\n";
  std::cout << get_cookie_time(t + 3600*24) << "\n";
}
--------------------------------
// 8桁のランダムな数字を文字列で返す関数
#include <random>
#include <sstream>
#include <iomanip>
#include <iostream>
#include <ctime>


std::string rand_n_digit()
{
   static std::default_random_engine dre((unsigned)time(0));
   std::uniform_int_distribution<int> di(1,99999999);
   std::stringstream ss;
   ss << std::setfill('0')<< std::setw(8) << di(dre) ;
   return ss.str();
}

int main()
{
   std::cout << rand_n_digit() << "\n";
   std::cout << rand_n_digit() << "\n";
   std::cout << rand_n_digit() << "\n";
}
--------------------------------
```

課題：セッション管理を行う検索ページ
ログイン画面と検索画面の2つのページからなるシステムを考える。
ユーザにユーザ名とパスワードを入力させるログイン画面を表示し、
これに入力すると検索画面に移行して郵便番号の検索ができる。
一定期間が過ぎると再度ログイン画面が表示されて、
ユーザ名とパスワードの入力を促すようなシステムを作りたい。
HTTP Cookieを用いて、この様なシステムを作成せよ。
話を単純にするために検索内容は郵便番号を入力すると
対応する住所を出力するものとする。

そのために、
CGI側では
	-getメソッドでクッキー無しまたはDBに一致しないクッキー付きの場合
		ログイン画面を表示
	-getメソッドで、DBに一致するクッキー付きの場合
		検索画面を表示
	-postメソッドで、クッキー無しまたはDBに一致しないクッキー付きの場合
		->ログイン画面を表示
	-postメソッドで、DBに一致するクッキー付きの場合
		pnumがなければ検索画面を表示
		pnumがあれば検索結果とともに検索画面を表示

	「DBに一致するクッキー」とは有効期限内のもののみとする

Webクライアント側
	ログイン画面に対して、
		ユーザ名とパスワードを入力させ、
		postメソッドでuser=XXX, pass=YYY としてCGIを起動する
	検索画面に対して、
		検索内容をpostメソッドで郵便番号をpnum=NNNNNNNとして
		CGIを起動する

Cookieの有効期限を1-3分程度にしてに試すこと。
データベースのスキーマは以下の様にすると良い。
```
	CREATE TABLE account (
	   user   text not null,
	   pass   text not null,
	   expire integer,
	   cookie text
	);
```
expireにはtime_t型の値をそのまま入れると良い。time_t型は符号無し整数で
1970-01-01 00:00:00 (UTC)からの秒数である。
DBにはinteger型で保存することで有効期限の判別が簡単になる。

```
// 以下ヒント

int main() {
   CGIinput tbl;
   cout << "Content-type: text/html; charset=UTF-8\r\n";

   const char *e = getenv("HTTP_COOKIE");
   if (e != nullptr) {
      // クッキーが送られてきた
      string s = e;
      bool valid = check_cookie(s); // DBを検索してsを探す
      if (!valid /* sがない || s が期限切れ */ ) return html_login();
      string pnum = tbl["pnum"]; // POSTメソッドのpnum指定を取り出す
      return html_search(pnum);
   }

   e = getenv("REQUEST_METHOD");
   if ((e != nullptr && string(e) == "POST") {
      // POSTメソッドだった
      string user = tbl["user"];
      string pass = tbl["pass"];
      if (user == "" || pass == "") return html_login();   // form入力エラー
      if (userがDBにない) create_account(user,pass);       // 新アカウント作成
      else if (userとpassがDBにある) update_cookie(user);  // cookieを更新
      else return html_login();                            // pass間違い
      return html_search();          // 新規またはlogin成功
   }

   return html_login();
}
```