Réseaux 1

TP 9 – Traitement des formulaires HTML

– CORRIGE –

Objectif : vous introduire au traitement des formulaires html

Note : vous placerez vos fichiers html et cgi sur nyx, dans un répertoire `~/web/rx1/TD9`

1. Récupérer les données transmises

Un petit formulaire d'essai

1.1. Construisez un petit formulaire d'inscription et qui demandera les informations suivantes :

- nom (20 caractères au maximum)
- age (entre 10 et 30)
- sexe (boutons radio : fille, garçon)
- adresse e-mail
- sport (cases à cocher : foot, judo, danse ..)
- catégorie (liste déroulante : benjamin, minime, cadet ..)

```
<html>
<body>
<h1>Petit formulaire d'inscription</h1>
<form name="F1">
   Nom : <input name='nom' size='30' maxlength='20'>
   Age : <input name='age' size='2'  maxlength='2'>
   <input type='radio' name='sexe' value='1'> garçon
   <input type='radio' name='sexe' value='2'> fille
   <br/>
   E-mail : <input name='email' size='60'><br/>
   A quels sports voulez-vous vous inscrire ?<br>
   <input type='checkbox' name='sport' value='foot'> foot<br>
   <input type='checkbox' name='sport' value='judo'> judo<br>
   <input type='checkbox' name='sport' value='danse'> danse<br>
   <br>
   Catégorie :
   <select name='categ'>
        <option value=''>sélectionnez votre catégorie</option>
        <option value='1'>benjamin</option>
        <option value='2'>minime</option>
        <option value='3'>cadet</option>
   </select>
   <input type='submit' value='Envoyer'>
</form>
</body>
</html>
```

1.2. Ecrire le programme CGI qui récupère les données transmises par le formulaire et les retourne dans une page html. Tester. Pour réaliser l'extraction et le décodage de la chaîne de données transmises, vous utilisez le programme `query.c` donné en Annexe.

Copiez les programmes util.c, query.c et le fichier util.h dans votre répertoire de travail

```
nyx$ cp /home/www/site.DEPTINFO/cgi-src/util.c  .
nyx$ cp /home/www/site.DEPTINFO/cgi-src/util.h  .
nyx$ cp /home/www/site.DEPTINFO/cgi-src/query.c  .
```

Compilez le programme query.c, et nommez le binaire query.cgi

```
nyx$ gcc query.c  util.c -o query.cgi
```

Pour tester appeler le programme query.cgi dans la balise <form> avec la méthode get
(observez dans l'url les paramètres envoyés par le navigateur)

```
<form name="F1" action='query.cgi' method='get' >
```

1.3. Ajouter en Javascript une fonction qui vérifie les conditions suivantes :

- tous les champs sont remplis
- l'âge est un entier compris entre 10 et 30
- l'adresse mail contient un @ et un « . »

Si ces conditions ne sont pas vérifiées, on refuse d'exécuter l'action associée au formulaire.

Pour vérifier le formulaire, on ajoute la fonction javascript VerifForm() dans l'en-tête du document

```
<head>
<script language="javascript">
function verifForm(f) {
   // verifier si tous les champs sont remplis
   if   (
             f.nom.value.length == 0 ||
             f.age.value.length == 0  ||
             ! ( f.sexe[0].checked || f.sexe[1].checked ) ||
             f.email.value.length == 0 ||
             ! ( f.sport[0].checked || f.sport[1].checked || f.sport[2].checked ) ||
             f.categ.value == ""
        ) {
        alert("Tous les champs n'ont pas été remplis !");
        return false ;
   }

   // verifier si l'age est un entier compris entre 10 et 30
   if ( ! VerifAge (f.age.value, 10, 30) ) {
        alert("L'age n'est pas numérique ou compris entre 10 et 30")
        return false ;
   }

   // verifier la présence des caractères @ et . dans mail
   if ( f.email.value.indexOf("@") < 0  || f.email.value.indexOf(".") < 0 ) {
        alert("mail incorrect")
        return false ;
   }

   return true;
}

function VerifAge(valeur, min, max) {
     for( var i=0; i< valeur.length; i++) {
          var ch= valeur.substring(i,i+1);
          if (ch < "0" || ch > "9") {
               return false;
          }
     }
     var nombre = parseInt (valeur, 10)
     if ( (nombre < min) || (nombre > max) ) {
            return false;
     }
     return true;
}
</script>
```

Pour bloquer l'appel au programme CGI en cas d'erreur, on utilise le gestionnaire d'évènement onSumit dans la balise <form>

```
<form name="F1" action='query.cgi' method='get' onSubmit='return verifForm(this);'>
```

2. Suivi de session à l'aide de champs cachés

Jeu du nombre secret

2.1. Ecrire un programme CGI en C qui propose de jouer au nombre secret.

Le programme a un nombre secret que l'utilisateur doit trouver. Ce dernier sait au départ dans quel intervalle se trouve ce nombre et il peut proposer à chaque tour de jeu un nombre. Si c'est le nombre secret, on lui indique qu'il a gagné, sinon on lui indique si le nombre proposé est trop petit ou trop grand par rapport au nombre secret. Programmer ce jeu, en supposant que c'est toujours le même nombre qui est à trouver.
Suggestion : utilisez un champ caché pour compter et mémoriser les nombres d'essais successifs.

```
/* programme nbsecret.c */
#include <stdio.h>
typedef struct {      /* structure (champ, valeur) */
   char name[128];
   char val[128];
} entry;
/* fonctions de util.c dans cgi-src de Apache sur nyx      */
```

```
void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

main(int argc, char *argv[]) {
   int min = 1 ;       /* nbre min                  */
   int max = 999 ;     /* nbre max                  */
   int secret = 314 ;  /* nbre secret               */
   int nombre = 0 ;    /* nbre propose              */
   int essai = 0 ;     /* nbre d'essai              */
   int trouve = 0 ;    /* 1 si trouve, 0 sinon      */

   /* tableau pour récupérer les couples champ, valeur du formulaire */
   entry entries[10];
   register int x,m=0;
   char *cl;

   /* on recupere les couples champ, valeur comme dans query.c  */
   cl = (char * ) getenv("QUERY_STRING");
   if(cl != NULL) {
        for(x=0;cl[0] != '\0';x++) {
             m=x;
             getword(entries[x].val,cl,'&');
             plustospace(entries[x].val);
             unescape_url(entries[x].val);
             getword(entries[x].name,entries[x].val,'=');
        }
        essai  = atoi(entries[0].val) ;
        nombre = atoi(entries[1].val) ;
   }

   /* on envoi le formulaire jeu du nombre secret                */
   printf("Content-type: text/html\n\n");

   printf("<html>\n");
   printf("<head>\n");
   printf("<title>Jeu du nombre secret</title>\n");
   printf("<script language=\"javascript\">\n");
   printf("  function verifNbre(valeur, min, max) {\n");
   printf("     var nombre = parseInt (valeur, 10)\n");
   printf("     if ( isNaN(nombre) || nombre < min || nombre > max ) {\n");
   printf("         alert(\"Donnez un nombre entre \"+min+\" et \"+max);\n");
   printf("         return false;\n");
   printf("     }\n");
   printf("     return true;\n");
   printf("  }  \n");
   printf("</script>\n");
   printf("</head>\n");

   printf("<body>\n");
   /* si premier appel, envoi page de bienvenue                          */
   if ( essai == 0 ) {
        printf("<p>Bienvenue au Jeu du nombre secret.</p>\n");
        printf("<p>Je pense à un nombre entre %d et %d </p>\n", min, max);
   }
   /* si appels suivants, envoi indication plus petit ou plus grand      */
   if ( essai != 0 ) {
        if ( nombre < secret )
             printf("<p>Votre nombre %d est <b>trop petit</b></p>\n",nombre);
        if ( nombre > secret )
             printf("<p>Votre nombre %d est <b>trop grand</b></p>\n",nombre);
        if ( nombre == secret ) trouve = 1;
   }
   /* si non trouve, envoi formulaire pour saisir un autre nombre        */
   /* le champ caché 'essai' permet de stocker le nombre d'essais        */
   if ( ! trouve ) {
        essai++ ;
        printf("<form action=\"nbsecret.cgi\" method=\"get\" \n");
        printf("nSubmit=\"return verifNbre(this.nombre.value,%d,%d);\">\n", min, max);
        printf("<input type=\"hidden\" name=\"essai\" value=\"%d\">\n", essai);
        printf("Proposez un nombre? <input type=\"text\" name=\"nombre\">\n");
        printf("<input type=\"submit\" value=\"Envoyer\">\n");
        printf("</form>\n");
   }

   if ( trouve ) {
        printf("<p>Bravo! vous avez trouvé en %d essais</p>\n", essai);
   }

   printf("</body>\n");
   printf("</html>\n");
}
```

3. Exercice de programmation

3.1. Il s'agit d'écrire un programme CGI en C qui sera utilisé à la fois pour créer et pour traiter un formulaire QCM sur des capitales.

Les questions seront stockées dans un fichier de texte Unix avec le format suivant : une ligne par question comportant le nom d'un pays suivi des capitales à choisir, suivi du numéro de la bonne réponse. Les champs sont séparés par un caractère « : »
Exemple :

```
Finlande:Oslo:Stockholm:Helsinki:Copenhague:3
Hongrie:Budapest:Varsovie:Prague:Bucarest:1
```

On supposera que pour chacune des questions, une seule réponse est possible (boutons radio).

Le programme lorsqu'il est appelé sans paramètre, crée le questionnaire et les propositions de réponses. L'utilisateur pourra envoyer le questionnaire et connaître ses résultats en cliquant sur un bouton, après avoir répondu à toutes les questions.

Le programme lorsqu'il est appelé par envoi du questionnaire, retourne les résultats. On retournera pour chacune des questions, un rappel de la question suivie de la liste des propositions en indiquant :
- la réponse correcte de couleur verte
- la réponse que l'utilisateur a sélectionnée de couleur verte suivi de BRAVO! si elle est correcte, et rouge sinon.

En fin des résultats le programme donne le nombre d'erreurs total.

Vous incorporez dans le formulaire une fonction JavaScript pour vérifier que l'utilisateur a répondu à toutes les questions, bloquer l'envoi si ce n'est pas le cas.

Le programme qcm.c proposée en solution dessous sert à la fois pour créer et pour traiter le formulaire QCM.
Pour savoir s'il doit créer ou traiter le formulaire, il contrôle si la chaîne contenant l'ensemble des couples (nom du champ, valeur) est vide ou non. La variable d'état « reponse » prend la valeur 0 ou 1 suivant le cas.

Le formulaire QCM est créé de la manière suivante :
- les questions sont nommées Q1, Q2, et ainsi de suite, énoncés
- les propositions de réponses d'une question sont représentées par un groupe de boutons radio
- le nom d'un groupe de boutons radio est le nom de la question correspondante
- la valeur d'un bouton radio d'un groupe est égale au rang de ce bouton dans le groupe
- la feuille de style qcm.css est utilisée pour les titres des questions (balises <h1>)
- la fonction JavaScript verifForm() dans qcm.js est utilisée pour vérifier si l'utilsateur a répondu à toutes les questions, et bloquer l'envoi du formulaire sinon
-

Exemple :

```
<html>
<head>
<title>QCM capitales</title>
<link rel="stylesheet" href="qcm.css" type="text/css">
<script langage="javascript" src="qcm.js"></script>
</head>
<body>
<form action="qcm.cgi" method="get" onSubmit="return verifForm(this);">
<h1>Question Q1 : Quelle est la capitale de la Finlande ?</h1>
<input type="radio" name="Q1" value="1">Oslo<br>
<input type="radio" name="Q1" value="2">Stockholm<br>
<input type="radio" name="Q1" value="3">Helsinki<br>
<input type="radio" name="Q1" value="4">Copenhague<br>
<h1>Question Q2 : Quelle est la capitale de la Hongrie ?</h1>
<input type="radio" name="Q2" value="1">Budapest<br>
<input type="radio" name="Q2" value="2">Varsovie<br>
<input type="radio" name="Q2" value="3">Prague<br>
<input type="radio" name="Q2" value="4">Bucarest<br>
<br/>
<input type="submit" value="Envoyer">
</form>
</body>
</html>
```

Le formulaire QCM est traité de la manière suivante :
- les données du formulaire QCM sont récupérées à l'aide des fonctions util.c de Apache, comme dans l'exemple query.c
- les propositions d'une question sont représentées par une liste d'items
- deux classes sont définies pour les propositions dans la liste d'items <li>: vrai ou faux
- la feuille de style qcm.css est utilisée pour définir les propriétés visuelles de ces deux classes

 # Programme qcm.c

```
#include <stdio.h>
#define QCM "qcm.dat"       /* nom du fichier de texte contenant le QCM   */
#define LG_MAX_LIGNE 120    /* nb max de caracteres par question          */
#define NB_MAX_CHOIX 20     /* nb max de choix par questions              */

typedef struct {
   char name[128];
   char val[128];
} entry;

/* fonctions util.c dans cgi-src de Apache  */
void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);

main(int argc, char *argv[]) {
   FILE      *f ;           /* fichier texte contenant le qcm      */
   char      ligne[LG_MAX_LIGNE+1] ;  /* ligne de texte           */
   /* tableaux pour récupérer les différents champs d'une question */
   char      *elem[NB_MAX_CHOIX+2];

   int  reponse=0;          /* 1 si reponse au QCM , 0 sinon                    */

   int  no_question=0 ;     /* no de question courante                          */
   int  ind_choix=0 ;       /* indice element courant de la question courante   */
   int  nb_choix=0 ;        /* nbre de choix de la question courante            */
   int  nb_erreurs=0 ;      /* nb de reponses erronnees                         */

   /* tableau pour récupérer les couples champ, valeur du formulaire */
   entry entries[10];
   register int x,m=0;
   char *cl;

   /* recuperer les couples champ, valeur, comme dans query.c  */
   reponse=0;
   cl = (char * ) getenv("QUERY_STRING");
   if(cl != NULL) {
        if (cl[0] != '\0' ) reponse=1;
        for(x=0;cl[0] != '\0';x++) {
             m=x;
             getword(entries[x].val,cl,'&');
             plustospace(entries[x].val);
             unescape_url(entries[x].val);
             getword(entries[x].name,entries[x].val,'=');
        }
   }

   /* envoi du QCM capitales */
   printf("Content-type: text/html\n\n");

   printf("<html>\n");
   printf("<head>\n");
   printf("<title>QCM capitales</title>\n");
   printf("<link rel=\"stylesheet\" href=\"qcm.css\" type=\"text/css\">\n");
   printf("<script langage=\"javascript\" src=\"qcm.js\"></script>\n");
   printf("</head>\n");
   printf("<body>\n");

   /* ouvrir le fichier */
   f = fopen (QCM, "r") ;

   if (reponse==0) {
        printf("<form action=\"qcm.cgi\" method=\"get\" \n");
        printf("onSubmit=\"return verifForm(this);\">\n");
   }

   /* on parcourt le fichier QCM  */
   no_question=0;
   while ( fgets(ligne,120,f) ) {
        /* on recupere les differents champs sur la ligne */
        ind_choix=0;
        elem[ind_choix] = (char *) strtok(ligne,":");
        while (elem[ind_choix]) {
             ind_choix++;
             elem[ind_choix] = (char *) strtok(NULL,":");
        }
        nb_choix=ind_choix-2;
        no_question++;

        /* envoi du QCM     */
        if (reponse==0) {
             printf("<h1>Question Q%d : Quelle est la capitale de la %s ?</h1>\n",
```

```
                              no_question, elem[0]);
            for (ind_choix=1; ind_choix<=nb_choix; ind_choix++) {
                 printf("<input type=\"radio\" name=\"Q%d\" value=\"%d\">%s<br>\n",
                              no_question, ind_choix, elem[ind_choix]);
            }
       }

       /* traitement du QCM     */
       if (reponse==1) {
            printf("<h1>Quelle est la capitale de la %s ?</h1>\n", elem[0]);
            printf("<ul>\n");
            for (ind_choix=1; ind_choix<=nb_choix; ind_choix++) {
                 if ( atoi(entries[no_question-1].val) == ind_choix
                      && ind_choix == atoi(elem[nb_choix+1]) ) // bravo bonne reponse
                      printf("<li class=\"vrai\">%s %s</li>\n",
                           elem[ind_choix], "BRAVO!");
                 if ( atoi(entries[no_question-1].val) == ind_choix
                      && atoi(elem[nb_choix+1]) != ind_choix ) { // mauvaise reponse
                      printf("<li class=\"faux\">%s</li>\n", elem[ind_choix]);
                      nb_erreurs++;
                 }
                 if ( atoi(entries[no_question-1].val) != ind_choix
                      && ind_choix == atoi(elem[nb_choix+1]) )    // bonne reponse
                      printf("<li class=\"vrai\">%s</li>\n", elem[ind_choix]);
                 if ( atoi(entries[no_question-1].val) != ind_choix
                      && ind_choix != atoi(elem[nb_choix+1]) )
                      printf("<li>%s</li>\n", elem[ind_choix]);
            }
            printf("</ul>\n");
       }
  }

  if (reponse==0) {
       printf("<br/>\n");
       printf("<input type=\"submit\" value=\"Envoyer\">\n");
       printf("</form>\n");
  } else {
       printf("<br/>\n");
       printf("Vous avez fait <b>%d erreurs</b>\n", nb_erreurs);
  }
  printf("</body>\n");
  printf("</html>\n");
}
```

Feuille de style qcm.css

```
h1 {    font-family: sans-serif  ; font-size: medium ;
        background-color: yellow ; color: blue ;    }

li.vrai { color: lime ;     }
li.faux { color: red ;      }
```

Fonction JavaScript pour vérifier si l'utilisateur a répondu à toutes les questions

```
// Fichier qcm.js
function verifForm(aForm) {
   var question_lu=""; // nom de la question suivante
   var question_cour="";    // nom de la question courante
   var a_repondu=false ;    // vrai si l'utilsateur a répondu à la question courante
   var i=0;

   // on regarde combien d'éléments il y a dans le formulaire
   // égal au nombre de boutons radio plus 1 (pour le bouton submit)
   n = aForm.length;

   // on fait une boucle pour examiner chaque question
   question_lu=aForm.elements[0].name;
   while (i < n-1) {
        // debut de traitement d'une question
        question_cour=aForm.elements[i].name;
        a_repondu=false;
        // on parcourt les boutons radio de la question courante (
        while (question_lu==question_cour) {
             if ( aForm.elements[i].checked ) a_repondu = true;
             i++;
             question_lu=aForm.elements[i].name;
        }
        // fin de traitement d'une question
        if ( ! a_repondu )  {
             alert("Vous n'avez pas répondu à la question "+question_cour);
             return false;
        }
   }
   return true;
```

```
}
```

4. Annexe – le programme query.c

Note : ce programme est fourni par Apache, il se trouve sur *nyx* dans `/home/www/site.DEPTINFO/cgi-src` (`query.c` utilise les fonctions de base contenues dans le programme `util.c`)

```
#include <stdio.h>
#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv();
#endif
#include <string.h>

typedef struct {
    char name[128];
    char val[128];
} entry;

void getword(char *word, char *line, char stop);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);


main(int argc, char *argv[]) {
    entry entries[100];
    register int x,m=0;
    char *cl;

    printf("Content-type: text/html%c%c",10,10);

    if(strcmp(getenv("REQUEST_METHOD"),"GET")) {
        printf("This script should be referenced with a METHOD of GET.\n");
        exit(1);
    }

    cl = getenv("QUERY_STRING");
    if(cl == NULL) {
        printf("No query information to decode.\n");
        exit(1);
    }
    for(x=0;cl[0] != '\0';x++) {
        m=x;
        getword(entries[x].val,cl,'&');
        plustospace(entries[x].val);
        unescape_url(entries[x].val);
        getword(entries[x].name,entries[x].val,'=');
    }

    printf("<H1>Query Results</H1>");
    printf("You submitted the following name/value pairs:<p>%c",10);
    printf("<ul>%c",10);

    for(x=0; x <= m; x++)
        printf("<li> <code>%s = %s</code>%c",entries[x].name,
                entries[x].val,10);
    printf("</ul>%c",10);
}
```