JRES 2001 - tutoriel n°1

Web dynamique : des pages aux applications

Pascal AUBRY
IFSIC - Université de Rennes 1
http://perso.ifsic.univ-rennes1.fr/aubry/presentations/jres2001



Introduction

Le protocole HTTP
Des pages statiques aux pages dynamiques



Le World Wide Web

- Système d'information hyper-média réparti
  - texte, images, images animées, son, vidéo …
- Informations stockées sur les serveurs
  - www.jres.org, www.ifsic.univ-rennes1.fr, www.ulaval.ca
- Interrogation des serveurs
  - tels que Apache, IIS, Netscape, Roxen, AOL, iPlanet, Lotus Domino, NCSA httpd, goAhead
  - par clients tels que Netscape, Mosaic, Internet Explorer, Lynx, Opera, Amaya



Les principes du Web

- Modèle client-serveur
- Le client envoie des requêtes au serveur
  - transfert de fichiers
  - exécution de programmes sur le serveur
  - mise à jour de fichiers
  - …
- Objets manipulés repérés par leur URL
- Utilisation du protocole HTTP



Le protocole HTTP

- Définit le langage utilisé pour les échanges entre client et serveur Web
  - version 0.9
    - simple protocole de transfert de données (GET et réponse)
  - version 1.0
    - restée un Internet Draft (RFC 1945)
  - actuellement version 1.1
    - RFC 2616 (juin 1999)
- Pas de session permanente entre client/serveur



Déroulement d'une requête HTTP

client — **protocole sans état** — serveur (démon HTTPD)

- Demande de connexion
- Attente de la réponse du serveur
- Établissement de la connexion
- Envoi d'une requête (URL)
- Réponse du serveur
- Affichage de la réponse
- Fermeture de la connexion



Exemple de transaction HTTP

```
% telnet www.ifsic.univ-rennes1.fr 80 ..... connexion au serveur web
Trying 148.60.4.30...
Connected to apollon.univ-rennes1.fr.
Escape character is '^]'.
GET /index.html HTTP/1.1 ..... demande de transfert
Host: apollon.univ-rennes1.fr ..... nom du serveur
From: pa@ifsic.univ-rennes1.fr ..... adresse demandeur (optionnelle)
                                    (ligne blanche = fin de l'entête HTTP de la requête)
HTTP/1.1 200 OK ..... réponse du serveur
Date: Tue, 02 Jun 2001 14:11:17 GMT
Server: Apache/1.3b6
Last-Modified: Mon, 07 Apr 2001 10:39:08 GMT ..... informations sur la ressource
ETag: "b3dd-524-33b78ccc"
Content-Length: 1316 ..... taille de la ressource
Accept-Ranges: bytes
Content-Type: text/html ..... type MIME
                                    (ligne blanche = fin de l'entête HTTP de la réponse)
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">      (contenu)
<HTML>
....
</HTML>
Connection closed by foreign host.          fermeture de la connexion
```


Les nouveautés de HTTP 1.1

- Identification obligatoire du *hostname* par le client
  - « *virtual hosting* »
- Négociation du contenu
  - documents français/anglais, HTML/PDF
- Connexions persistantes
  - amélioration temps de chargement de pages avec images
- Prise en charge des mandataires (*proxies*)
- Nouveaux codes réponse
- Nouvelles méthodes

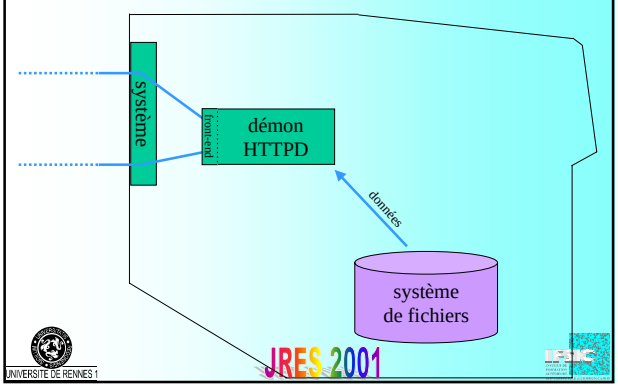
Les méthodes HTTP

- Demander une ressource
  - **GET** : rapatrier une ressource
  - **POST** : envoyer des données et rapatrier une ressource
- Avoir de l'information sur une ressource
  - **HEAD** : connaître ses caractéristiques
  - **OPTIONS** : connaître les options qui lui sont applicables
- Mettre à jour une ressource à distance
  - **PUT** : la créer ou remplacer son contenu
  - **DELETE** : la détruire
- Débogguer
  - **TRACE** : tracer les mandataires (*proxies*)

Les codes de réponse sous HTTP

- 10x : information
- 20x : succès — 200 OK
- 30x : redirection — 301 Moved permanently
- 40x : erreur du client — 401 Unauthorized, 402 Payment required, 403 Forbidden, 404 Not found
- 50x : erreur du serveur — 500 Internal server error

Délivrer un document statique



Le web à l'IFSIC

- Un réseau hétérogène
  - DOS, Windows, Solaris, Irix, Linux
- Des besoins applicatifs
  - configuration des comptes utilisateurs
  - changement des mots de passe
  - interrogations de quotas (disque, impression)
  - rendus des TPs aux enseignants
  - gestion administrative
    - édition de l'offre de formation
    - gestion des services et des heures complémentaires
- Un client « universel »

Pourquoi développer sur le web ?

- Universalité du protocole HTTP
  - indépendance vis-à-vis des clients
  - pérennité des applications
- Simplicité de l'interface
- Parce que ça fait moderne ;-)

Web dynamique : qui fait quoi ?

- Le serveur exécute, le client reçoit
  - SSI, XSSI, CGI, FastCGI, PHP, ASP, JSP
  - indépendance vis-à-vis du client (navigateur)
  - interactivité limitée
- Le serveur envoie, le client exécute
  - JavaScript embarqué (DHTML), Applet Java
  - dépendance vis-à-vis du client
  - plus d'interactivité

Pages dynamiques : solutions côté serveur

Le protocole CGI

FastCGI, Perl, PHP, Servlets, JSP, ASP

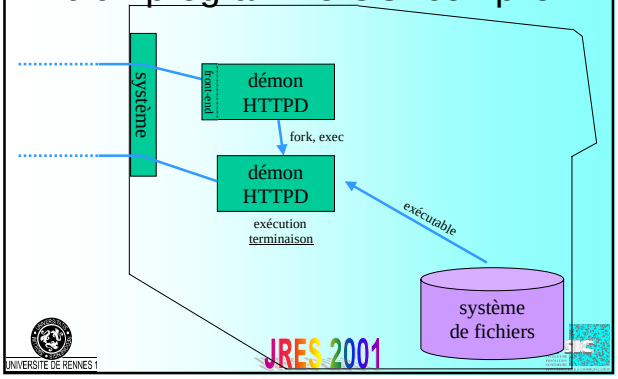
SSI, XSSI : le premier pas vers la dynamique

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head><title>Exemple XSSI</title></head>
<body bgcolor="#FFFFFF">
<!--#config errmsg="erreur de syntaxe dans un (x)SSI" -->
<!--#set var="x" value="test"--><!--#echo var="x"--><br>
Fichier modifié le : <!--#echo var="LAST_MODIFIED" --> <br>
Nom du serveur : <!--#echo var="SERVER_NAME" --> <br>
<!--#if expr="$HTTP_USER_AGENT = 'Mozilla/2.0 (compatible;
MSIE 3.01; Windows 95)'" -->
 Il est temps de se mettre à jour !
<!--#endif -->
</body>
</html>
```

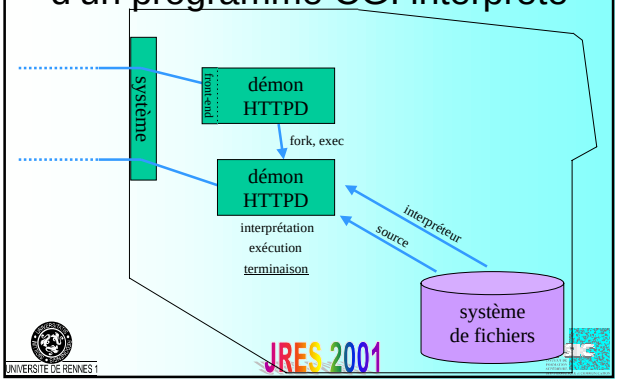

CGI : Common Gateway Interface

- Un standard pour l'interface entre applications et serveurs d'informations
- Permet de passer des paramètres aux requêtes
  - dans l'URL avec la méthode GET
    http://serv.dom.org/cgi-bin/script?arg1=val1&arg2=val2
  - comme des données avec la méthode POST
- Exécution d'un programme sur le serveur
  - Les informations renvoyées au client sont statiques
  - Des requêtes successives permettent le dynamisme

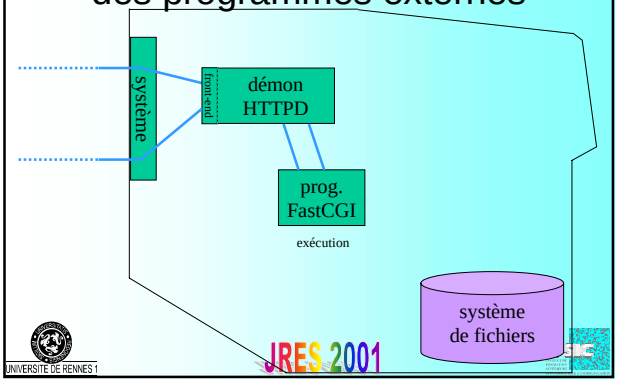
Exécution via HTTP d'un programme CGI compilé



Exécution via HTTP d'un programme CGI interprété



FastCGI : une solution au coût des programmes externes



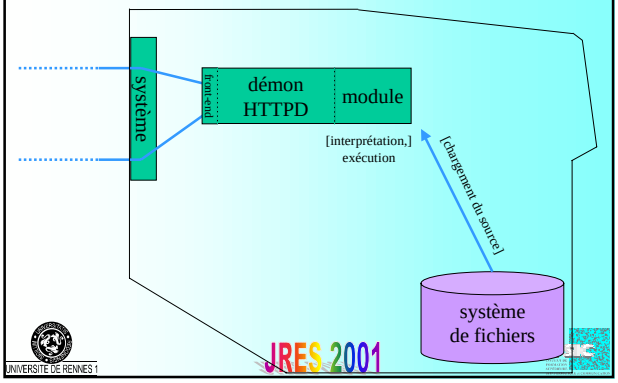
FastCGI

- Performance, simplicité (comme CGI)
- Migration facile (de l'existant CGI)
- Isolation du démon httpd
  - sécurité en cas de crash
  - Indépendance de l'architecture du serveur
- Distribution possible de la charge

Scripts côté serveur : les quatre leaders du marché

- Perl
- PHP (Php Hypertext Preprocessor)
- JSP (Java Server Pages)
- ASP (Active Server Pages)

L'approche modulaire (PHP, Perl)



Perl

- Practical Extraction and Report Language
- Ressources illimitées
- Module d'Apache (mod_perl) ou stand-alone
- Multi-plateformes
- On aime ou on n'aime pas ;-)

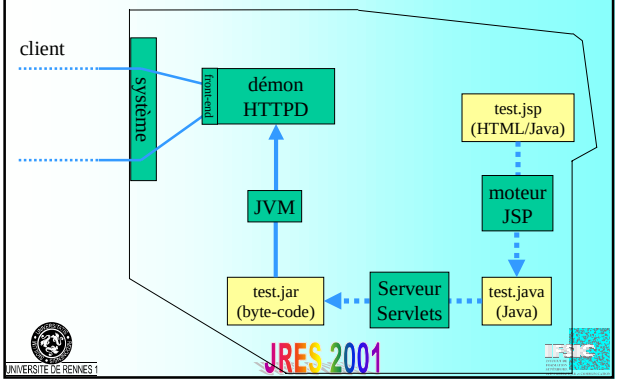
PHP

- Multi-plateformes
- Le plus simple
- Un langage non généraliste (comme Java et Perl) mais néanmoins très riche (bibliothèques)
- Un produit libre

Java

- Les servlets
  - Programme Java exécuté côté serveur
  - Transformation en byte-code avant exécution
  - Servies par un serveur dédié (ex : Tomcat)
- JSP (Java Server Pages)
  - Pages HTML avec Java embarqué
  - Transformation en Servlets

Le mécanisme JSP



ASP

- Tourne nativement sur IIS (MicroSoft)
- JavaScript ou VBScript
- ASP+ :
  - support langages compilés (VB, C++, C#)
  - compilation intermédiaire JS et VBS
  - mécanisme de cache des objets compilés
  - configuration format XML
  - contrôle accru des formulaires
- portabilité limitée (www.chilisoft.com)

Les langages de CGI

- Les shells
  - sh, tcsh, bash
- Les langages compilés
  - C, C++, Pascal, ...
- Perl
- Java (JSP, Servlets)
- PHP
- Python
- VBScript, JavaScript (ASP)

Un site web : des responsabilités, des métiers

- Technique (ingénieur)
  - fonctionnement du service
  - mise à disposition d'outils
  - développement
- Éditoriale (rédacteur)
  - mise à jour du contenu
- Graphique (infographiste)
  - présentation du site
- Pénale (juriste)

Pourquoi un environnement de développement ?

- Pour gérer l'accès concurrent
  - les droits sur les systèmes de fichiers sont insuffisants
  - on utilise souvent une base de données et/ou une interface web
- Pour séparer les tâches
  - graphisme, interfaçage Homme/Machine
  - développement
  - mise à jour du contenu
- Pour intégrer plusieurs technologies
  - COM, CORBA, EJB, C/C++, Java, XML

Les environnements de développement

- BEA WebLogic (Java)
- Lutris Enhydra (Java, open source)
- Netscape Application Server
- Jbuilder (Java)
- Zope (Python)
- Midgard (PHP, open source)
- Allaire/Macromedia ColdFusion 5 (CFML)
- Vignette Content Suite V6
- ...

Une comparaison rapide…

	[Java|VB]Script (+ASP)	Java (+JSP)	Perl (+mod_perl)	PHP	Python (+Zope)
Installation	★★★★★	★★	★★★★★	★★★★★	★★★★★
Apprentissage	★★★	★★	★	★★★★★	★★★★
Puissance	★★★	★★★★★	★★★	★★★	★★★
Portabilité	★	★★★★★	★★★★★	★★★	★★★
Outils	★★★★	★★★★	★★	★★★	★★★
Ressources	★★★	★★★★	★★★★★	★★★★★	★★★

…et subjective !

Comment choisir ?

- 1. Évaluer le ou les projets à mettre en œuvre
- 2. Évaluer ses moyens
  - humains
  - financiers
- 3. Tenir compte de l'existant
  - les habitudes
  - les compétences
  - les préférences
- 4. Se faire soi-même une idée des produits

Les problèmes de la programmation dynamique sur le web à travers PHP

PHP en quelques mots

- Un langage
  - qui comprend CGI
  - complet
  - simple
  - hérité de Perl, C et sh
- Un module d'Apache
  - performant
  - existe aussi en stand-alone
  - en pleine évolution
    - au départ pour les pages personnelles ;-)
    - aujourd'hui pour les applications sur le web

Un programme PHP, c'est…

- Un fichier HTML…
- dans lequel on trouve des tags PHP :

```
<html>
<head><title>TEST</title></head>
<body>
   <p>
      Il est <?echo date("H:i");?>.
   </p>
</body>
</html>
```


Dans la pratique… (surtout pour les gros projets)

- Un programme PHP est un ensemble d'instructions PHP qui affichent :
  - du code HTML
  - ou autre chose :
    - ascii
    - PostScript, PDF
    - GIF, JPEG, PNG, ...
- L'approche « pages » est désormais supplantée par l'approche « composants »

Une page HTML avec des instructions PHP

```
<html>
  <head>
    <title>
      Bonjour !
    </title>
  </head>
  <body>
    <p>
      sur le serveur, il est exactement
      <? echo date("H:i:s") ; ?>
    </p>
  </body>
</html>
```

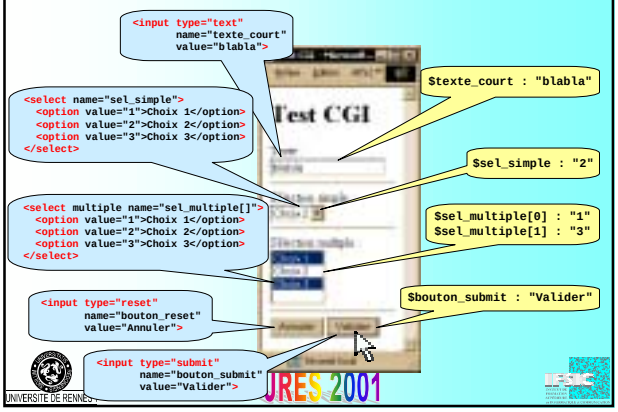

Un programme PHP

```
class sortie {
  function sortie($titre)  // constructeur
    { $this->titre = $titre ; }
  function debut()
    { echo "<html><head>$this->titre</head><body>" ; }
  function fin()
    { echo "</body></html>" ; }
}

    $s = new sortie("Bonjour !") ;
    $s->debut() ;
    echo "<p>sur le serveur, il est exactement "
        .date("H:i:s")
        ."</p>" ;
    $s->fin() ;
```


Les paramètres CGI

```
<form action="test_cgi.php" method="GET">
  Texte :<br>
  <input type="text" name="texte_court" value="blabla"><hr>
  Sélection simple :<br>
  <select name="sel_simple">
    <option value="1">Choix 1</option>
    <option value="2">Choix 2</option>
    <option value="3">Choix 3</option>
  </select><hr>
  Sélection multiple :<br>
  <select multiple name="sel_multiple[]">
    <option value="1">Choix 1</option>
    <option value="2">Choix 2</option>
    <option value="3">Choix 3</option>
  </select><hr>
  <input type="reset" name="bouton_reset" value="Annuler">
  <input type="submit" name="bouton_submit" value="Valider">
</form>
```


Les paramètres CGI



PHP et les bases de données

- Un support très large
  - Adabas D, dBase, filePro, Hyperwave, Informix, InterBase, mSQL, MS SQL Server, MySQL, Oracle 7, Oracle 8, SyBase, PostgreSQL, Solid, ODBC
- Utiliser la bibliothèque de son SGBD
- En l'absence de bibliothèque, utiliser ODBC
  - ou développer sa propre bibliothèque ;-)
- Il existe des interface génériques
  - phpDB

Connexions persistantes (aux bases de données)

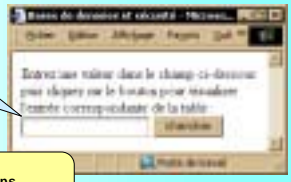
- La connexion à une base de données est souvent l'opération la plus coûteuse d'une requête
- Identification d'une connexion : (*host*, *database*, *user*, *password*)
- Au lieu d'ouvrir et fermer une connexion à une base de données à chaque requête, on peut garder les connexions ouvertes et les mémoriser pour pouvoir les réutiliser par la suite

Accès aux bases de données

```
Require("DB.php") ;

$conn = DB::Connect("mysql://user:passwd@host/db" ) ;

$res = $conn->Query(" . . . ") ;

while ( $row = $res->FetchRow() )
    { . . . }

$conn->Close() ;
```


Sécurité et bases de données

```
<form ...>
    ...
    <input type="text"
           name="cle">
    <input type="submit"
           name="go"
           value="chercher">
</form>
```



```
$req_sql =
    "SELECT DesChamps
     FROM UneTable
     WHERE cle = '$cle' ";
```

J'essaierai bien la clé suivante :

```
4' ;
SELECT * FROM … ;
DROP DATABASE … ;
SELECT '1
```


Sécurité et bases de données

- Tout ce qui provient de l'utilisateur doit être considéré comme suspect :
  - paramètres CGI, fichiers téléchargés, ...

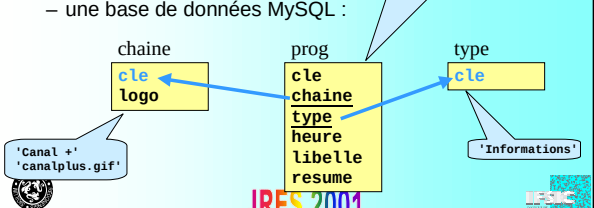
```
$req_sql = "
    SELECT DesChamps FROM UneTable
    WHERE cle = '". Addslashes($cle) ."'" ;
```

DAMNED !

Zappeur.org

- Objectif : faire un site web d'interrogation des programmes de télévision
- Moyens :
  - un serveur Web (RedHat Linux 7.1, Apache 1.3.19, [illegible] .0.4pl1)
  - une base de données MySQL :

```
1458
'Canal +'
'Informations'
'2001-12-09 12:40:00'
'Le vrai journal'
'Présenté par Karl Zéro...'
```



Récupération des types d'émission

```
// importation de la bibliothèque DB
require_once( 'DB.php' );

// informations de connexion
$info = "mysql://zappeur:pass@localhost/zappeur" ;

// connexion à la base de données
$db = DB::connect($info) ;

// interrogation de la base de données
$sql = "SELECT cle FROM type ORDER BY cle" ;
$result = $db->query($sql) ;

// parcours du résultat
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
  { // $row["cle"] contient le type }

  // libération des ressources
  $result->free() ;
```


Récupération des programmes

```
$user_date = addslashes($annee-$mois-$jour) ;
$sql = "
SELECT cle, chaine, libelle, type,
       HOUR(heure) as h, MINUTE(heure) as m
    FROM programme
    WHERE heure >= '$user_date'
      AND heure < DATE_ADD('$user_date', INTERVAL 1 DAY)" ;
if ( !empty($type) )
  $sql .= " AND type = '".addslashes($type)."'" ;
$sql .= " ORDER BY heure,chaine" ;
$result = $db->query($sql) ;
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
  {
    // $row["chaine"], $row["h"], $row["m"],
    // $row["libelle"], $row["type"]
  }
$result->free() ;
```


Première ébauche

```
<?
  require_once( 'DB.php' );
  $titre ="zappeur.org" ;
  $db = DB::connect( "mysql://zappeur:pass@localhost/zappeur" );
?>

<html>
  <head><title><? echo $titre ; ?></title></head>
  <body>
   <h1><? echo $titre; ?></h1>
   <form action=<? echo $PHP_SELF ;?> method="POST">
     date : <input type="text" name="jour" size="2">
     / <input type="text" name="mois" size="2">
     / <input type="text" name="annee" size="4">
    <select name="type">
      <option value="">-- tous --</option>
      <?
        $result = $db->query("SELECT cle FROM type ORDER BY cle") ;
        while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
          echo "<option value=\"".$row["cle"]."\">".$row["cle"]."</option>" ;
        $result->free() ;
      ?>
    </select>
    <input type="submit" name="go" value="chercher">
   </form>
   <hr>
   <table border="1">
    <tr><th>Chaine</th><th>Horaire</th><th>Emission</th><th>Type</th></tr>
    <?
      $user_date = addslashes($annee)."-".addslashes($mois)."-".addslashes($jour) ;
      $sql = "SELECT cle, chaine, libelle, type,
              HOUR(heure) as h, MINUTE(heure) as m
              FROM programme
              WHERE heure >= '$user_date'
                AND heure < DATE_ADD('$user_date', INTERVAL 1 DAY)" ;
      if ( !empty($type) ) $sql .= " AND type = '".addslashes($type)."'" ;
      $sql .= " ORDER BY heure,chaine" ;
      $result = $db->query($sql) ;
      while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
          echo "<tr><td>".$row["chaine"]."</td>
                    <td>".$row["h"].":".$row["m"]."</td>
                    <td>".$row["libelle"]."</td>
                    <td>".$row["type"]."</td></tr>" ;
    ?>
   </table>
  </body>
</html>

<? $db->disconnect() ; ?>
```

Mémorisation de la date

- Pour que l'utilisateur n'ait pas à entrer la date lors de sa prochaine requête, on peut pré-remplir le formulaire de réponse avec les paramètres fournis

```
date : <input type="text" name="jour" size="2"
               value="<?echo $jour?>">
/      <input type="text" name="mois" size="2"
               value="<?echo $mois?>">
/      <input type="text" name="annee" size="4"
               value="<?echo $annee?>">
```


Mémorisation du type d'émission

- De la même manière on peut mémoriser le type d'émission souhaité par l'utilisateur

```
$result = $db->query("SELECT cle FROM type ORDER BY cle") ;
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
    echo "<option value=\"".$row["cle"]."\""
        .( ($row["cle"]==$type) ? " selected" : "" )
        .">".$row["cle"]."</option>" ;
$result->free() ;
```


Contrôle des paramètres

- La date fournie par l'utilisateur étant par nature non fiable, il faut :
  - vérifier qu'elle correspond bien à quelque chose
  - éventuellement l'initialiser

```
$jour = intval($jour) ;
$mois = intval($mois) ;
$annee = intval($annee) ;
if ( !checkdate($mois,$jour,$annee) )
    list($jour,$mois,$annee) = array( date("d"),
                                      date("m"),
                                      date("Y")) ;
```

La forme et le fond

- On veut séparer le contenant du contenu
  - pour pouvoir changer le « look » de l'application sans modifier l'application elle-même
  - pour faire travailler différentes personnes (développeurs, infographistes) en même temps sur un projet
- Les solutions
  - utiliser des feuilles de style (CSS, pas toujours suffisant)
  - adopter un environnement de développement qui offre cette possibilité (Zope, Midgard, …)
  - se servir de « templates »

Création d'un template

```
<html>
  <head><title>__TITRE__</title></head>
  <body>
    <h1>__TITRE__</h1>
    <form action=__ACTION__ method="POST">
      date : <input type="text" name="jour" size="2" value="__JOUR__">
      / <input type="text" name="mois" size="2" value="__MOIS__">
      / <input type="text" name="annee" size="4" value="__ANNEE__">
      <select name="type">
        <option value="">-- tous --</option>
        __TYPES__
      </select>
      <input type="submit" name="go" value="chercher">
    </form>
    <hr>
    <table border="1">
     <tr><th>Chaîne</th><th>Horaire</th><th>Émission</th><th>Type</th></tr>
      __RESULTATS__
    </table>
  </body>
</html>
```

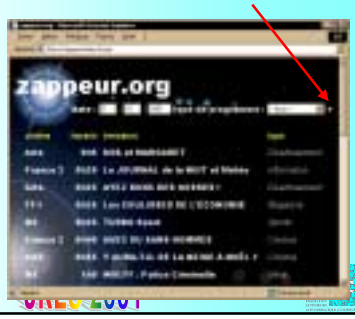

Utilisation d'un template

```
function affichage($titre,$jour,$mois,$annee,$types,$resultats)
{
  // lecture du template dans un tableau
  foreach ( file("modeles/modele4.html") as $ligne )
    {
      // replacement des tokens du template par les valeurs à afficher
      $ligne = str_replace("__TITRE__",$titre,$ligne) ;
      $ligne = str_replace("__ACTION__",$GLOBALS["PHP_SELF"],$ligne) ;
      $ligne = str_replace("__JOUR__",$jour,$ligne) ;
      $ligne = str_replace("__MOIS__",$mois,$ligne) ;
      $ligne = str_replace("__ANNEE__",$annee,$ligne) ;
      $ligne = str_replace("__TYPES__",$types,$ligne) ;
      $ligne = str_replace("__RESULTATS__",$resultats,$ligne) ;
      echo $ligne ;
    }
}
```


Place aux infographistes...

- Une image cliquable pour soumettre les requêtes :

```
<input type="image"
       src="images/submit.png"
```



- Un fond d'écran créé avec Gimp
- Des polices et des couleurs agréables

Plus d'interactivité : JavaScript

- Code embarqué dans les pages HTML...
- …donc exécuté par le client
  - attention à la portabilité !
- Programmation objet événementielle
  - onLoad, onClick, onFocus, onSubmit, …
- Dynamic HTML

JavaScript : exemple 1

```
<html>
  <head>
    <title>Identifiez-vous !</title>
    <script language="JavaScript">
      function TestNom()
      { if (document.ident.nom.value != "")
          return true ;
        alert("Merci de vous identifier avant de continuer.") ;
        return false ;
      }
    </script>
  </head>
  <body>
    <h1>Identifiez-vous !</h1>
    <form action="ident.php"
          name="ident"
          onSubmit="return TestNom()">
      <input type="text" name="nom">
      <input type="submit" name="go" value="continuer">
    </form>
  </body>
</html>
```

JavaScript : exemple 2

```
<html>
  <head>
    <title>Aide</title>
    <script language="JavaScript">
      function Bouton()
      { var popup =window.open("aide.php",
                               "aide_popup",
                               "width=300,height=300");
          popup.Focus() ; }
    </script>
  </head>
  <body>
    <h1>Aide : cliquez sur le bouton</h1>
    <form>
      <input type="button" name="go" value="aide"
             onClick="Bouton() ; return false">
    </form>
  </body>
</html>
```

Régler la date facilement

- Une image cliquable :

```
<a href="#" onClick="date_plus();return false">
  <img src="images/up.png" border="0">
</a>
```

- Une fonction Javascript :

```
function date_plus()
{
  var jour = window.document.criteres.jour.value ;
  var mois = window.document.criteres.mois.value ;
  var annee = window.document.criteres.annee.value ;
  jour ++ ;
  if ( jour == 32 )
    { jour = 1 ; mois ++ ;
      if ( mois == 13 )
        { mois = 1 ; annee ++ ; } }
  window.document.criteres.jour.value = jour ;
  window.document.criteres.mois.value = mois ;
  window.document.criteres.annee.value = annee ;
}
```

Rafraîchir la page à chaque réglage

```
<select name="type"

  onChange="window.document.criteres.submit()">


function date_plus()
  {
    /* . . . */
    window.document.criteres.submit() ;
  }
```

Utiliser deux cadres différents

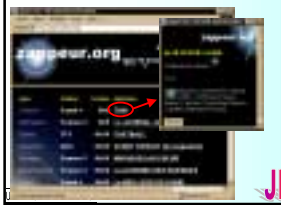
```
<head>
  <frameset rows="150,*">

    <frame name="haut"
           src="__ACTION__?cadre=haut"
           scrolling="no"
           noresize>

    <frame name="bas"
           src="__ACTION__?cadre=bas">

    <noframes>
      Pas de frame, pas de programme !
    </noframes>
  </frameset>
</head>
```

`top.haut.document.criteres.jour.value`

Afficher plus d'informations

- Une fonction JavaScript...

```
function popup(prog)
{ var popup_win =  window.open(
                     '__ACTION__?cadre=popup&prog='+prog,
                     'zappeur_popup',
                     'location=no,toolbar=no') ;
  popup_win.focus() ; }
```



- … déclenchée par un lien :

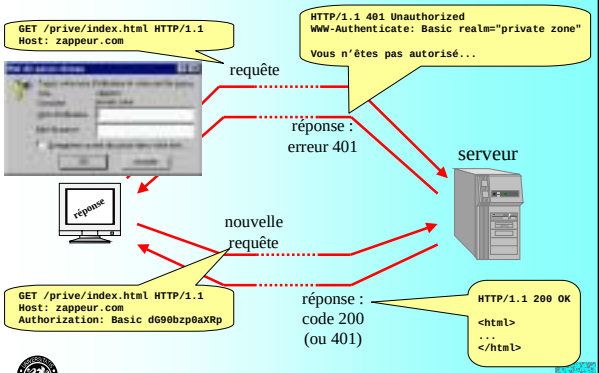
```
<a href="#"
   onClick="popup('$cle');
            return false">
   $libelle
</a>
```

La consécration : zappeur.com

L'authentification sur le web

- Transport des informations d'authentification
  - protocole HTTP (royaume)
  - cookies
  - SSL (Secure Socket Layout)
- Vérification
  - par le démon HTTP ou par l'application
- Référence
  - fichiers, base de données, codage « en dur »

Authentification sous HTTP



Vérification de l'authentification

- Par le démon httpd
  - authentification HTTP simple
    - sur un fichier de mots de passe (htpasswd)
    - sur une base de données (mod_auth_mysql)
    - sur un annuaire (auth_ldap)
  - authentification SSL
    - sur des certificats (X509)
- Par applicatif
  - sur n'importe quoi !

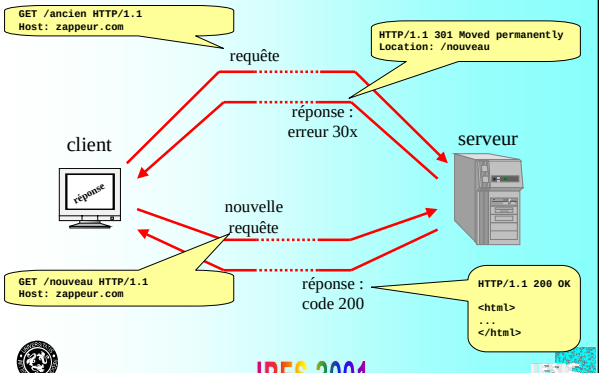
Protéger le site avec HTTP

```
if (    ($PHP_AUTH_USER != "titi")
     || ($PHP_AUTH_PW != "toto") )
{
  header('WWW-Authenticate: Basic realm="zappeur"');
  header("HTTP/1.0 401 Unauthorized");
  echo "Vous n'êtes pas autorisé...\n" ;
  exit ;
}

// à partir d'ici, l'utilisateur est authentifié
```


Protéger le site avec HTTPS

- Par le démon httpd
  - vérification du certificat client suivant configuration
- Par applicatif
  - protocole
    - `$SLL_PROTOCOL, $SSL_CIPHER, $SSL_CIPHER_ALGKEYSIZE,`
  - vérification du certificat client
    - `$SSL_CLIENT_CERT, $SLL_CLIENT_CERT_CHAINx, $SSL_CLIENT_VERIFY`
  - informations sur les certificats client et serveur
    - `$SSL_XXX_I_DN_C, $SSL_XXX_I_DN_CN, $SSL_XXX_I_DN_Email, $SSL_XXX_I_DN_L, $SSL_XXX_I_DN_O, $SSL_XXX_I_DN_OU, $SSL_XXX_I_DN_ST, $SSL_XXX_M_SERIAL, $SSL_XXX_M_VERSION, $SSL_XXX_S_DN, $SSL_XXX_S_DN_C, $SSL_XXX_S_DN_CN, $SSL_XXX_S_DN_Email,`

Redirection sous HTTP



Forcer le protocole HTTPS (redirection immédiate)

```
If ( $HTTPS != "on")
{
  header("Location: https://$SERVER_NAME:443$REQUEST_URI");
  exit ;
}

// à partir d'ici, on sait que l'on est sous HTTPS
```

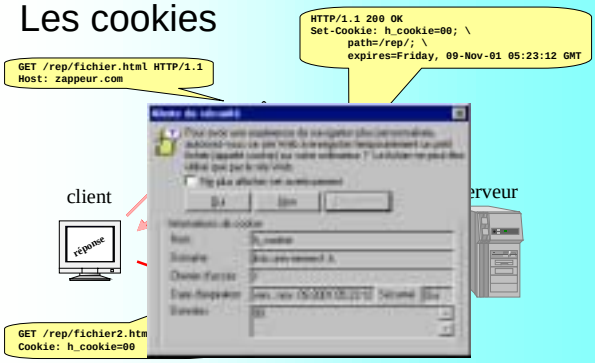

Forcer le protocole HTTPS (redirection différée)

```
If ( $HTTPS != "on")
  {
    $new_url = "https://$SERVER_NAME:443$REQUEST_URI" ;
    header("Refresh: 2;url=$new_url");
    echo "<h2>zappeur.com utilise maintenant
          le protocole HTTPS</h2>
          Vous allez être redirigé vers l'URL
          suivante dans quelques instants :
          <a href=\"$new_url\">$new_url</a>." ;
    exit ;
  }
```

Les cookies

- Ce sont des lignes particulières de l'entête HTTP qui permettent de véhiculer des informations entre client et serveur
- Caractéristiques :
  - nom (**`UTILISATEUR`**)
  - valeur (**`Pascal.Aubry`**)
  - date de validité (**`Sunday, 04-Nov-01 23:12:40 GMT`**)
  - domaine/serveur (**`www.zappeur.com`**)
  - chemin (**`/public`**)

Les cookies



Les cookies (limitations)

- La taille
  - 4Ko par cookie
- Le nombre
  - 300 cookies par navigateur
  - 20 cookies par serveur (pour un même client)
- Les clients peuvent « refuser » des cookies

Mémoriser l'horaire préféré d'un utilisateur entre ses visites

- On envoie un cookie qui va (peut-être) être stocké sur le client et renvoyé à chaque requête

```
// on prend l'ancienne valeur si $h est vide
if ( empty($h) && !empty($h_cookie) )
  { $h = $h_cookie ; }
/* ... */
// on renvoie la nouvelle valeur si elle a changé
if ( $h != $h_cookie )
  SetCookie(
      "h_cookie",                  // nom
      $h,                          // valeur
      time()+604800,               // validité
      "/",                         // chemin
      ".ifsic.univ-rennes1.fr") ; // domaine
      1) ;                         // HTTPS
```

Transmettre des variables d'état

- Objectif : propager des variables entre les requêtes
  - identité du visiteur
  - informations de connexion
  - ...
- Le problème
  - HTTP est un protocole sans état
- Les solutions
  - paramètres CGI cachés
  - cookies
  - sessions

Solution 1 : paramètres CGI cachés

- Dans chaque formulaire et chaque lien hypertexte, on ajoute un (des) paramètre(s) caché(s) :

```
<form ...>
   <input type="hidden" name="ident" value="durand">
   <input type="hidden" name="pass" value="xx98yy">
   ...
</form>
<a href="action.php?ident=durand&pass=xx98yy">texte</a>
```

- Problèmes :
  - Les liens hypertextes et les grosses variables d'état
  - on transmet toutes les variables d'état à chaque requête
  - ça marche, mais ça va bien un moment… ;-)

Solution 2 : cookies

- À chaque changement d'une variable d'état, on envoie un cookie au client qui pourra nous le renvoyer lors de la requête suivante
- Intérêts :
  - on ne transmet les variables que lorsqu'elles changent
  - on ne fait pas de ré-écriture des formulaires et des liens
- Problèmes :
  - nombre et longueur des cookies
  - il faut encore le faire « à la main »
  - certains ne supportent pas les cookies...

Solution 3 : les sessions

- Affectation d'un ID unique
  - pour chaque visiteur non connu (sans ID)
  - de forme aléatoire
- Liaison (identifiant - données sur le serveur)
  - stockage mémoire, disque, base de données, ...
  - format propriétaire, XML (WDDX), ...
- Transmission de l'identifiant
  - par cookie, ré-écriture (automatique)

Transmission des IDs de session

- Ré-écriture automatique :
  -
```
<form …>
   <input type="hidden"
          name="SESS_ID"
          value="5kj81l12yhs3">
   ...
</form>
```
  - `<a href="…?…&SESS_ID=5kj81l12yhs3">`
  - `<img src="…?…&SESS_ID=5kj81l12yhs3">`
  - `<frame src="…?…&SESS_ID=5kj81l12yhs3">`
  - ...

Tracer la connexion des utilisateurs

```
// on active le mécanisme de session
session_start() ;
// on déclare les variables de session
session_register("nb_req") ;
session_register("debut") ;

$nb_req++ ;
if ( empty($debut) ) $debut = time() ;
// on calcule le temps de connexion
$temps = time() - $debut ;
```

Télécharger des fichiers sur le serveur (uploading)

- Il est parfois intéressant de transmettre les données sous forme de fichiers
  - données de taille importante
  - données déjà stockées sous forme de fichier sur le client
- Un encodage particulier du formulaire
- Une prise en charge par le serveur

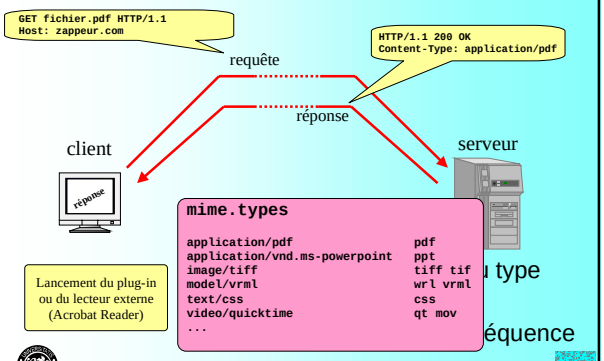
Formulaire de téléchargement

```
<form enctype="multipart/form-data"
      action="upload.php"
      method="post">
  <input type="hidden"
         name="MAX_FILE_SIZE"
         value="50000">
  Envoyer ce fichier :
  <input name="fichier" type="file">
  <br>
  <input type="submit"
         value="Envoyer">
</form>
```

Accepter un téléchargement

```
// test pour vérifier que l'utilisateur
// a bien transmis un fichier
if ( $HTTP_POST_FILES["fichier"]["tmp_name"] == "none" )
  return ;

// sauvegarde sous le nom original
// (nom du fichier sur le client)
if ( !move_uploaded_file(
                $tmp_name,
                "repert/". $HTTP_POST_FILES["fichier"]["name"]
                ))
  return ;

// taille du fichier : $HTTP_POST_FILES["fichier"]["size"]
...
```

Y'a pas que HTML sur le web !



Créer des images dynamiques

- avec la librairie GD

```
// ouverture de l'image
if (!($im = ImageCreateFromPNG("logos/img")) )
  return ;
// allocation de la couleur d'écriture
$fg = ImageColorAllocate ($im, 255, 255, 255);
// affichage d'un texte
ImageTTFText($im,16,0,10,20,$fg,"comicbd.ttf","texte") ;
// envoi sur la sortie standard
header("Content-Type: image/jpeg") ;
ImageJpeg($im) ;
ImageDestroy($im) ;
```

Renvoyer les programmes au format XML

```
Header("Content-Type: text/xml") ;

echo '<?xml version="1.0" encoding="ISO-8859-1"?>' ;
echo '<racine>' ;
/* … */
echo '</racine>' ;
```

- En bref, on peut renvoyer n'importe quoi !

Une connexion dure...

- … au moins un certain temps !
- Largement le temps pour qu'elle soit interrompue
  - par le client (arrêt du chargement en cours)
  - par le serveur (défaillance quelconque)
  - par le média (panne sur le réseau)
- Il faut s'y préparer
  - transactions avec des bases de données
  - …

JRES 2001

Alors, qu'attendez-vous ?

http://perso.ifsic.univ-rennes1.fr/aubry/presentations/jres2001

JRES 2001

zappeur.com

Un exemple d'application PHP d'interrogation de base de données

1. Présentation

Ce document décrit la mise en œuvre d'un site web présentant à ses visiteurs les programmes télé. Il ne s'agit bien évidemment ici que d'un exemple de mise en œuvre, tant les sites de ce genre sont légion sur le web. Notre but est de montrer comment on peut construire un tel site progressivement, étape par étape.

Mise en place du serveur

La machine utilisée pour héberger le site sera un PC, sur lequel on installe la distribution Linux 7.1 de RedHat (configuration serveur web).
Sont installés en particulier les packages suivants :

- Apache 1.3.19 (le serveur web)
- Php 4.0.4pl1 (le langage de script)
- MySql 3.23.36 (le SGBD)

Le serveur est appelé `zappeur`. Une fois l'installation terminée, on démarre les services MySql (base de données) et http (serveur web).

Stockage des données

Les données de l'application sont stockées dans une base de données MySql. La structure des données est la suivante :

Table **`chaine`** (chaînes de réception)

- `cle` (`text`) : le nom de la chaîne (ex : « `France 2` »)
- `logo` (`text`) : le nom du logo sur le système de fichier (ex : « `fr2.gif` »)

Table **`type`** (types de programme)

- `cle` (`text`) : le nom du type de programme (ex : « `Sports` »)

Table **`programme`** (émissions)

- `cle` (`integer`) : index de la table (ex : « `1` »)
- `chaine` (`text`) : chaîne sur laquelle passe le programme (ex : « `France 2` », référence à la table `chaine`)
- `type` (`text`) : type de programme (ex : « `Sports` », référence à la table `type`)
- `heure` (`datetime`) : heure à la quelle passe le programme
- `libelle` (`text`) : intitulé du programme (ex : « `Tout le sport` »)
- `resume` (`text`) : informations supplémentaires (ex : « `présenté par G. Holtz` »)

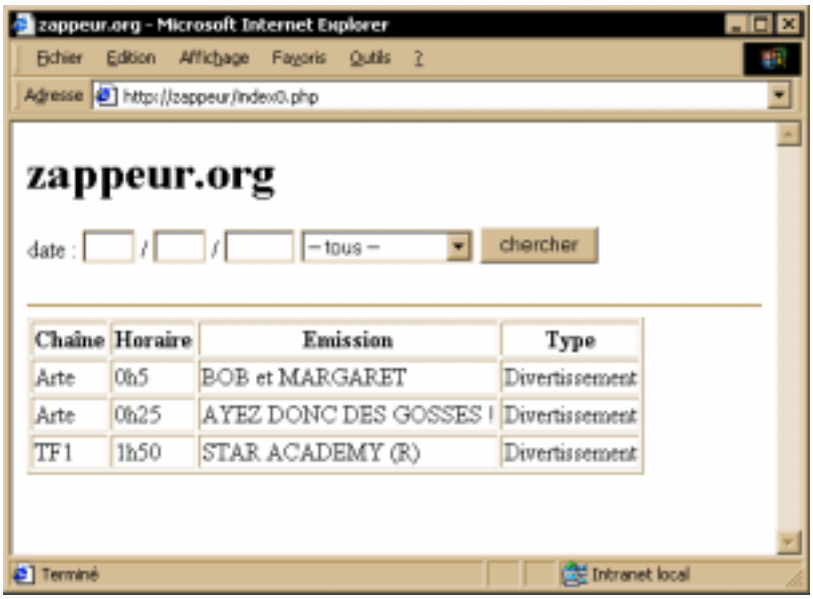
Les données de la base sont initialisées par un script, qui effectue les opérations suivantes :

- Création d'une base de données (`zappeur`) ;
- Création d'un utilisateur (`zappeur`) protégé par mot de passe (`pass`) ;
- Autorisation de lecture pour l'utilisateur `zappeur` sur toute la base `zappeur` ;

- Création des entrées de la base.

2. Création d'une ébauche

On désire obtenir une page basique composée :
- d'un formulaire (en haut) grâce auquel l'utilisateur pourra entrer la date et le type des programmes qu'il désire consulter ;
- des résultats correspondant à sa recherche.



On crée une page HTML dans laquelle on insère les instructions suivantes :
- Appel de la librairie PhpLib DB (`require_once('DB.php'));`
- Définition du titre (`$titre = …`)
- Connexion à la base de données (`$db = DB::connect(…)`)
- Affichage du titre (`echo $titre`)
- Affichage du nom du script (`echo $PHP_SELF`)
- Affichage du choix du type de programme (`$result = …`)
- Affichage des programmes correspondant aux choix de l'utilisateur (`$user_date = …`)
- Déconnexion de la base de données (`$db->disconnect()`)

```
<?
// première ébauche de l'application
  require_once( 'DB.php' );
  $titre ="zappeur.org" ;
  $db = DB::connect( "mysql://zappeur:pass@localhost/zappeur" );
?>

// (suite sur la page suivante)
```

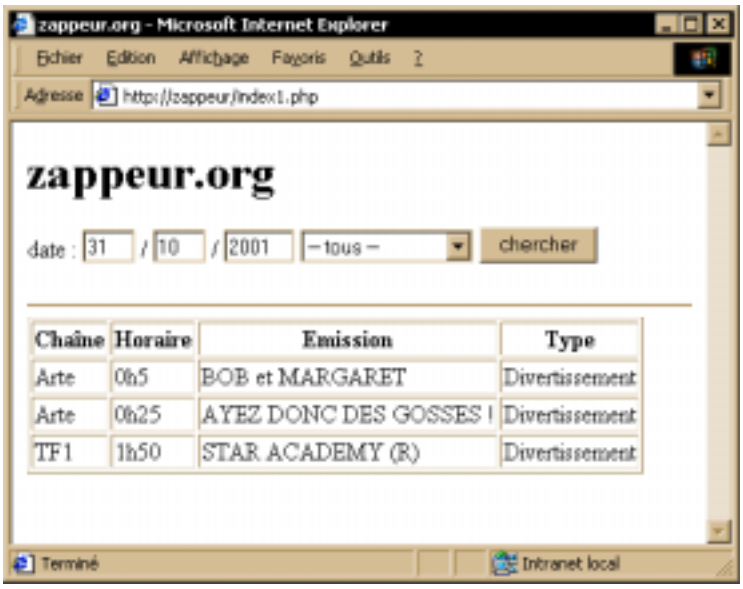
```
<html>
  <head><title><? echo $titre ; ?></title></head>
  <body>
    <h1><? echo $titre; ?></h1>
    <form action=<? echo $PHP_SELF ;?> method="POST">
      date : <input type="text" name="jour" size="2">
      / <input type="text" name="mois" size="2">
      / <input type="text" name="annee" size="4">
      <select name="type">
        <option value="">-- tous --</option>
        <?
          $result = $db->query("SELECT cle FROM type ORDER BY cle") ;
          while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
            echo "<option value=\"".$row["cle"]."\">".$row["cle"]."</option>" ;
          $result->free() ;
        ?>
      </select>
      <input type="submit" name="go" value="chercher">
    </form>
    <hr>
    <table border="1">
      <tr><th>Chaîne</th><th>Horaire</th><th>Emission</th><th>Type</th></tr>
      <?
        $user_date = addslashes($annee)."-".addslashes($mois)."-".addslashes($jour) ;
        $sql = "SELECT cle, chaine, libelle, type, HOUR(heure) as h, MINUTE(heure) as m
                FROM programme
                WHERE heure >= '$user_date'
                  AND heure < DATE_ADD('$user_date', INTERVAL 1 DAY)" ;
        if ( !empty($type) )
          $sql .= " AND type = '".addslashes($type)."'" ;
        $sql .= " ORDER BY heure, chaine" ;
         echo $sql ;
        $result = $db->query($sql) ;
        while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
          {
              echo "<tr>
                    <td>".$row["chaine"]."</td>
                    <td>".$row["h"]."h".$row["m"]."</td>
                    <td>".$row["libelle"]."</td>
                    <td>".$row["type"]."</td></tr>" ;
        }
      ?>
    </table>
  </body>
</html>
<? $db->disconnect() ; ?>
```

Avec ce script, l'utilisateur entre la date dans les champs prévus à cet effet, et les programmes correspondant à la date entrée, s'affichent dans la partie inférieure de la fenêtre du navigateur.

3. Mémorisation de la date donnée par l'utilisateur

Avec le script précédent, si l'utilisateur veut effectuer une nouvelle requête par exemple en changeant le type de programme désiré, il doit ressaisir la date. Afin d'améliorer l'ergonomie, on mémorise cette date en remplissant les champs du formulaire proposé avec les valeurs donnés par l'utilisateur lors de la requête précédente :

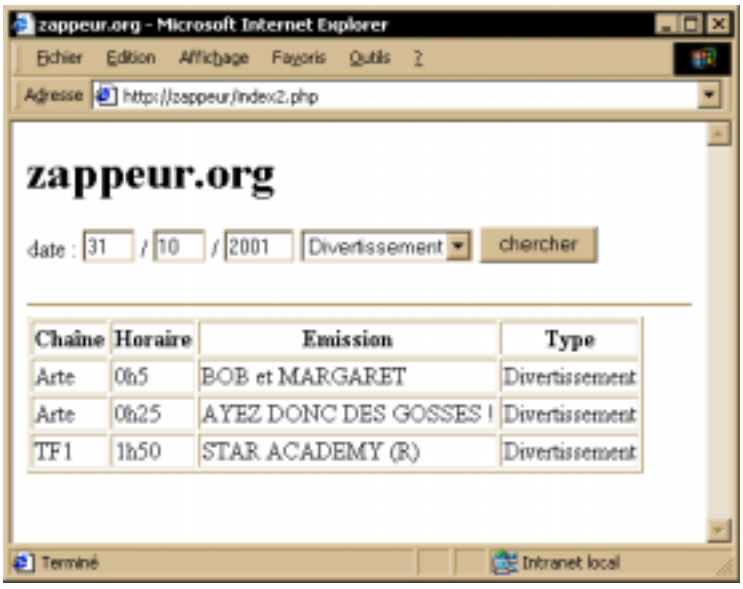
```
date : <input type="text" name="jour" value="<? echo $jour; ?>" size="2">
      / <input type="text" name="mois" value="<? echo $mois; ?>" size="2">
      / <input type="text" name="annee" value="<? echo $annee; ?>" size="4">
```



4. Mémorisation du type d'émission

De la même manière, avec un peu plus de code, on peut mémoriser le type d'émission souhaité par l'utilisateur ; il s'agit d'indiquer comme « sélectionnée » l'option donnée par l'utilisateur :

```
$result = $db->query("SELECT cle FROM type ORDER BY cle") ;
while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
    echo "<option value=\"".$row["cle"]."\""
        .( ($row["cle"]==$type) ? " selected" : "" )
        .">".$row["cle"]."</option>" ;
$result->free() ;
```



5. Contrôle des paramètres

Les paramètres donnés par l'utilisateur sont par nature non fiables. Ils doivent systématiquement être contrôlés avant d'être utilisés. Ainsi dans le script précédent, l'utilisateur peut par mégarde entrer une chaîne de caractères dans un des champs de saisie de la date. En contrôlant les paramètres, on évite ce genre d'erreurs qui peuvent entraîner des dysfonctionnements de l'application :

```
$jour = intval($jour) ;
$mois = intval($mois) ;
$annee = intval($annee) ;
if ( !checkdate($mois,$jour,$annee) )
    list($jour,$mois,$annee) = array(date("d"),date("m"),date("Y")) ;
```

De cette manière, on évite les mauvaises saisies, et on initialise la date courante par défaut. Un utilisateur arrivant pour la première fois sur le site voit alors directement les programmes du jour.

6. Séparation de la forme et du fond

Un problème fréquent lors du maintien d'une application est la séparation du contenu et du contenant, c'est-à-dire les informations et la manière dont elles sont présentées. Une solution est l'utilisation de feuilles de style (CSS), mais cette méthode n'est pas toujours suffisante ; nous montrons ici l'utilisation d'un « *template* ».

Pour cela, nous séparons dans le script ce qui est de la forme (le code HTML) de ce qui est de l'application (le code PHP) en remplaçant le code PHP par des tags spéciaux :

```
<html>
  <head><title>__TITRE__</title></head>
  <body>
    <h1>__TITRE__</h1>
    <form action="__ACTION__" method="POST">
      date : <input type="text" name="jour" value="__JOUR__" size="2">
      / <input type="text" name="mois" value="__MOIS__" size="2">
      / <input type="text" name="annee" value="__ANNEE__" size="4">
      <select name="type">
        <option value="">-- tous --</option>
                    __TYPES__
      </select>
      <input type="submit" name="go" value="chercher">
    </form>
    <hr>
    <table border="1">
      <tr><th>Chaîne</th><th>Horaire</th><th>Emission</th><th>Type</th></tr>
                   __RESULTATS__
    </table>
  </body>
</html>
```

La fonction `affichage` *parse* le template HTML et remplace les tags.

```
function affichage($titre,$jour,$mois,$annee,$types,$resultats)
{
  foreach ( file("modeles/modele4.html") as $ligne )       // lecture du template dans un tableau
    {
      $ligne = str_replace("__TITRE__",$titre,$ligne) ;
      $ligne = str_replace("__ACTION__",$GLOBALS["PHP_SELF"],$ligne) ;
      $ligne = str_replace("__JOUR__",$jour,$ligne) ;
      $ligne = str_replace("__MOIS__",$mois,$ligne) ;      // on remplace les tags
      $ligne = str_replace("__ANNEE__",$annee,$ligne) ;    // par les valeurs à afficher
      $ligne = str_replace("__TYPES__",$types,$ligne) ;
      $ligne = str_replace("__RESULTATS__",$resultats,$ligne) ;
      echo $ligne ;
    }
}
```

7. Mise en place du graphisme

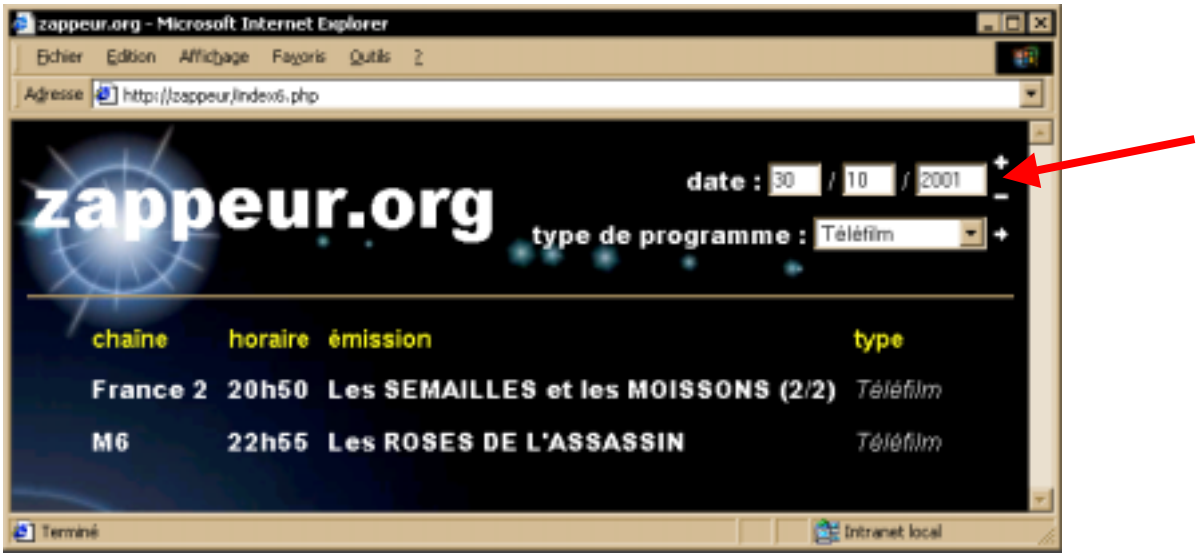
Si la transformation précédente n'a aucune conséquence immédiate pour l'utilisateur, elle permet de donner le *template* à un infographiste, qui pourra le transformer pour obtenir un *look* plus agréable.

8. Réglage plus facile de la date

La simple utilisation des formulaires HTML et du protocole HTTP (avec CGI) n'amène pas toujours l'ergonomie que l'on peut attendre d'une application grand public. Ce manque d'ergonomie peut en général être diminué à l'aide de code JavaScript embarqué dans les pages HTML.

Dans l'application que nous mettons en place, on peut par exemple éviter à l'utilisateur d'entrer la date à l'aide de chiffres en lui donnant deux images (« + » et « - ») lui permettant d'augmenter ou de diminuer la date affichée.

```
<head>
  <title>__TITRE__</title>
  <script language="javascript" src="javascript/date6.js"></script>
</head>
```



```
<a href="#" onClick="date_plus();return false"><img src="images/up.png" border="0"></a>
<a href="#" onClick="date_moins();return false"><img src="images/down.png" border="0"></a>
```

Le code JavaScript des deux fonctions `date_plus` et `date_moins` montre comment on peut mettre à jour les champs d'un formulaire depuis une fonction JavaScript.

```
function date_plus()
{
  var jour = window.document.criteres.jour.value ;
  var mois = window.document.criteres.mois.value ;
  var annee = window.document.criteres.annee.value ;
  jour ++ ;
  if ( (jour == 32)
    || ( (jour == 31) && ( (mois == 4) || (mois == 6) || (mois == 9) || (mois == 11) ) )
    || ( (jour == 30) && (mois == 2) && ((annee % 4) == 0) )
    || ( (jour == 29) && (mois == 2) && ((annee % 4) != 0) ) )
    {
      jour = 1 ;
      mois ++ ;
      if ( mois == 13 )
        { mois = 1 ; annee ++ ; }
    }
  // mise à jour des champs des formulaires
  window.document.criteres.jour.value = jour ;
  window.document.criteres.mois.value = mois ;
  window.document.criteres.annee.value = annee ;
}

function date_moins()
{
  var jour = window.document.criteres.jour.value ;
  var mois = window.document.criteres.mois.value ;
  var annee = window.document.criteres.annee.value ;
  jour -- ;
  if ( jour == 0 )
    {
      mois -- ;
      if ( mois == 0 )
        { mois = 12 ; annee -- ; }
      if ( (mois == 2) && ((annee % 4) == 0) )                              jour = 29 ;
      else if (mois == 2)                                                   jour = 28 ;
      else if ( (mois == 4) || (mois == 6) || (mois == 9) || (mois == 11) ) jour = 30 ;
      else                                                                  jour = 31 ;
    }
  window.document.criteres.jour.value = jour ;
  window.document.criteres.mois.value = mois ;
  window.document.criteres.annee.value = annee ;
}
```

9. Rafraîchissement automatique

Pour faciliter encore la consultation des programmes, on fait en sorte que les programmes correspondant à la sélection de l'utilisateur soient rafraîchis automatiquement après un changement des critères de sélection (date ou type de programme), sans avoir à cliquer après sur une image ou un bouton « *submit* ».

```
  <head>
    <title>__TITRE__</title>
    <script language="javascript" src="javascript/date7.js"></script>
    <script language="javascript" src="javascript/type7.js"></script>
  </head>

              type de programme :
              <select name="type" onChange="type_change()">
                <option value="">-- tous --</options>
                __TYPES__
              </select>

function type_change()
{ window.document.criteres.submit() ; }
```

```
function date_plus()
{
  /* ... */
  window.document.criteres.submit() ;
}

function date_moins()
{
  /* ... */
  window.document.criteres.submit() ;
}
```

10. Introduction de cadres (« *frames* »)

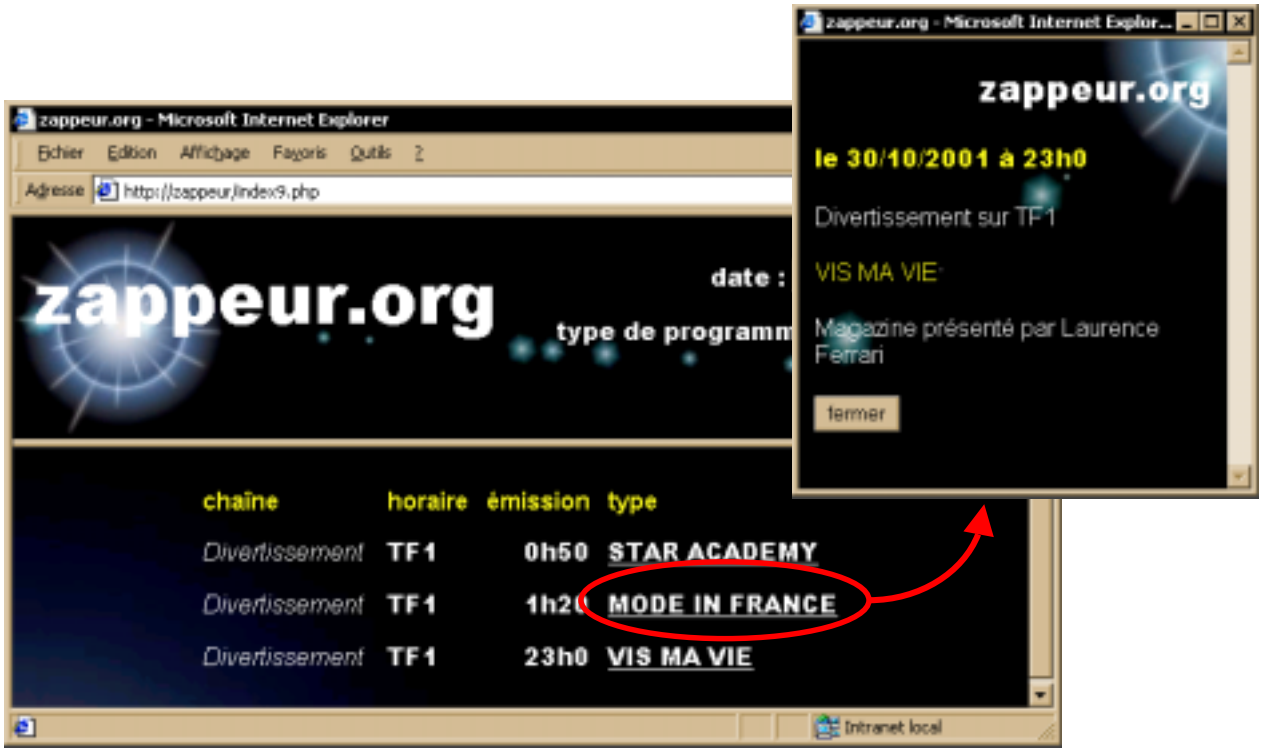
Dans la version précédente de notre application, le rafraîchissement de toute la fenêtre du navigateur, donc en particulier des critères de sélection, introduit un scintillement à chaque changement des critères de sélection. On peut se passer de ce scintillement en scindant la fenêtre du navigateur en deux cadres, deux « *frames* ». Seul le cadre du bas est rafraîchi lors des changements des critères de sélection.


```
<frameset rows="150,*">
  <frame name="haut" src="__ACTION__?cadre=haut" scrolling="no" noresize></frame>
  <frame name="bas" src="__ACTION__?cadre=bas"></frame>
  <noframes>Pas de frame, pas de programme !</noframes>
</frameset>
```

```
switch ($cadre)
  {
  case "haut":
    $db = db_connect() ;
    verif_param($jour,$mois,$annee) ;
    affichage_haut($titre,$jour,$mois,$annee,types($db,$type)) ;
    $db->disconnect() ;
    break ;
  case "bas":
    $db = db_connect() ;
    verif_param($jour,$mois,$annee) ;
    affichage_bas($titre,$jour,$mois,$annee,resultats($db,$type,$jour,$mois,$annee),$type) ;
    $db->disconnect() ;
    break ;
  default:
    affichage($titre) ;
  }
```

11. Ajouts des informations supplémentaires des émissions

Pour que l'utilisateur puisse consulter les informations supplémentaires sur les émissions sélectionnées, un lien est ajouté sur chaque émission affichée dans la cadre inférieur. Un clic sur ces liens fait apparaître une nouvelle fenêtre (« popup ») dans lequel s'affichent toutes les informations sur l'émission choisie.




```
$str .= "
  <tr>
    <td><font face=\"Arial\"><font color=\"#FFFFFF\"><em>".$row["type"]."</em></font></font></td>
    <td><font face=\"Arial Black\"><font color=\"#FFFFFF\">".$row["chaine"]."</font></font></td>
    <td align=right><font face=\"Arial Black\"><font color=\"#FFFFFF\">
      ".$row["h"]."h".$row["m"]."
    </font></font></td>
    <td><font face=\"Arial Black\"><font color=\"#FFFFFF\">
      <a href=\"#\" onClick=\"popup('".$row["cle"]."');return false\">".$row["libelle"]."</a>
    </font></font></td>
  </tr>" ;
```

```
switch ($cadre)
  {
  case "haut":
    /* . . . */ break ;
  case "bas":
    /* . . . */ break ;
    . . .
    break ;
  case "popup":
    $db = db_connect() ;
    affichage_popup($db,$titre,$prog) ;
    $db->disconnect() ;
    break ;
  default:
    affichage($titre) ;
  }
```

12. La consécration : zappeur.com

Pour améliorer la consultation des programmes, on ajoute un critère supplémentaire (l'heure de début de programme) sur le même modèle que les précédents. Il ne restait plus qu'à changer le nom du site puisque le domaine zappeur.org est déjà réservé, à afficher le logo des chaînes et de notre sponsor, et afficher une horloge permanente rappelant aux visiteurs du site qu'ils sont au boulot et qu'ils ont sûrement autre chose à faire que regarder les programmes télé…

13. Authentification HTTP

Pour diverses raisons (dont notre passage dans le monde .com ;-), on peut avoir besoin de limiter l'accès du site à certains privilégiés. Une possibilité est alors de mettre en place une protection basé sur HTTP (authentification basique).

```
$user = "toto" ;
$password = "titi" ;
if (   ($PHP_AUTH_USER != $user)
    || ($PHP_AUTH_PW != $password)
   )
  {
    header('WWW-Authenticate: Basic realm="zappeur"');
    header("HTTP/1.0 401 Unauthorized");
    echo "Pour entrer dans cette zone, vous devez être <b>$user</b> (mot de passe :
<b>$password</b>).\n" ;
    exit ;
  }
// à partir d'ici, l'utilisateur est authentifié
```

14. Forçage du protocole HTTPS (redirection immédiate)

La protection par HTTP valant ce qu'elle vaut (c'est-à-dire pas grand chose), on désire maintenant mettre en œuvre HTTPS. Si cela ne pose pas de problème du point de vue de la configuration du serveur (Apache est préconfiguré), on désire en revanche que les visiteurs puissent se connecter en HTTP au serveur, et soient automatiquement redirigés vers la version sécurisée du site.

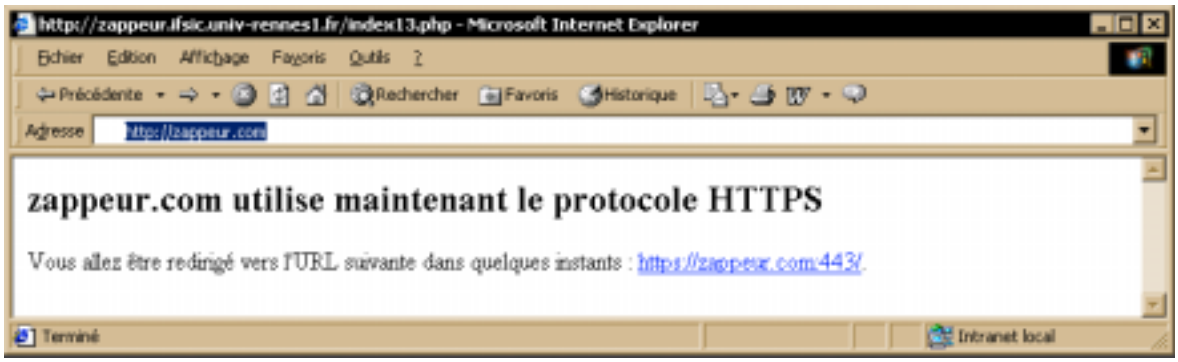
On utilise pour cela une redirection :

```
If ( $HTTPS != "on")
  {
    header("Location: https://$SERVER_NAME:443$REQUEST_URI");
    exit ;
  }
// à partir d'ici, on sait que l'on est sous HTTPS
```

15. Forçage du protocole HTTPS (redirection différée)

Si l'on souhaite prévenir les utilisateurs de la redirection, on peut utilisé un rafraîchissement différé vers la nouvelle URL :

```
If ( $HTTPS != "on")
  {
    $new_url = "https://$SERVER_NAME:443$REQUEST_URI" ;
    header("Refresh: 2;url=$new_url");
    echo "<h2>zappeur.com utilise maintenant le protocole HTTPS</h2>
          Vous allez être redirigé vers l'URL suivante dans quelques instants :
          <a href=\"$new_url\">$new_url</a>." ;
    exit ;
  }
// à partir d'ici, on sait que l'on est sous HTTPS
```



16. Mémorisation de l'horaire préféré de l'utilisateur (entre ses visites du site)

On veut maintenant que le visiteur, lors qu'il arrive sur le site, trouve l'horaire désiré déjà rempli avec la valeur qu'il avait donnée lors de sa visite précédente.
On va pour cela utiliser un *cookie*, valeur stockée sur le client qui sera repassée au serveur lors des visites suivantes.

```
function verif_param(&$h,$h_cookie,&$jour,&$mois,&$annee)
{
    /* vérification de la date */
    if ( empty($h) && !empty($h_cookie) )
        $h = $h_cookie ;
    $h = intval($h) ;
    if ( ($h < 0) || ($h > 24) )
        $h = date("H") ;
    format_chiffre2($h) ;
    if ( $h != $h_cookie )
        SetCookie("h_cookie",                // nom
                  $h,                        // valeur
                  time()+604800,             // validité
                  "/",                       // chemin
                  ".ifsic.univ-rennes1.fr",  // domaine
                  1) ;                       // HTTPS
}
```

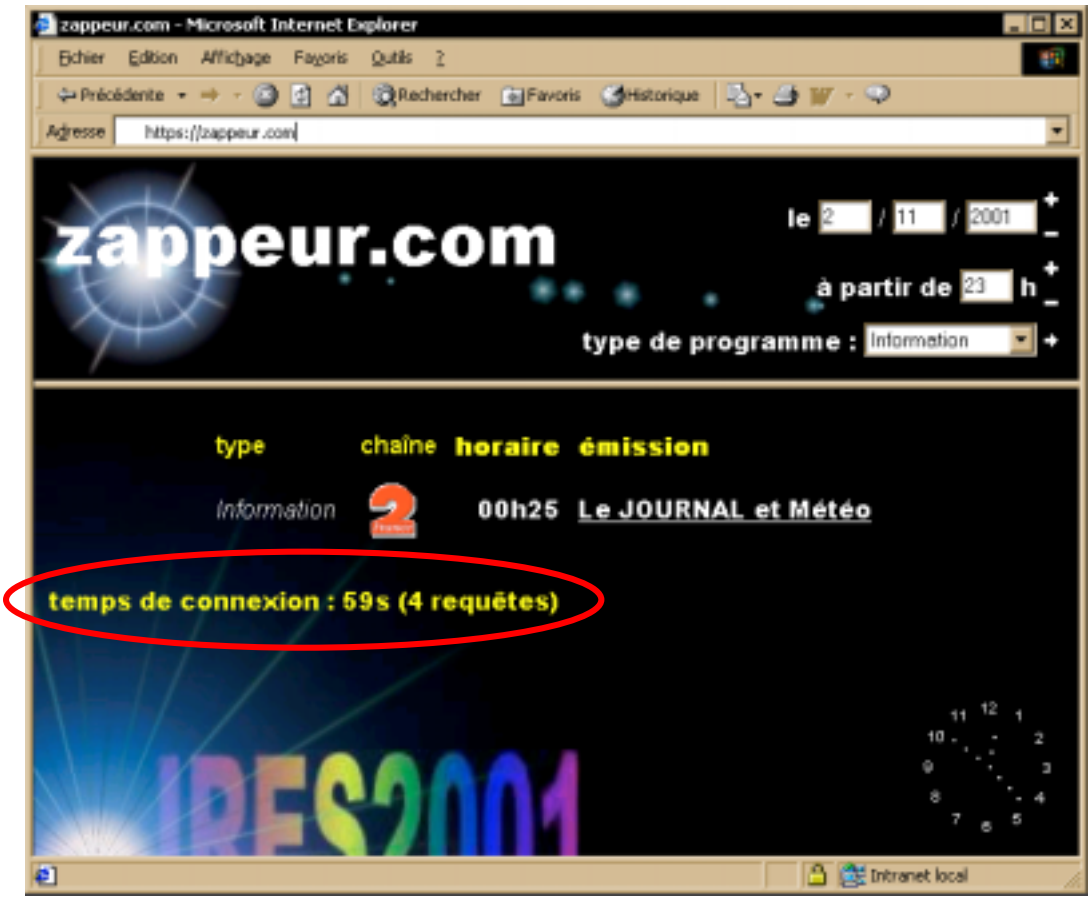
17. Affichage du temps de connexion

On souhaite maintenant afficher le temps de connexion d'un utilisateur, ainsi que le nombre de requêtes qu'il a effectué lors de sa visite. On pourrait pour cela utiliser, comme précédemment deux *cookies*. Connaissant les limitations des *cookies*, nous préférons utiliser le mécanisme de sessions de PHP. Deux variables de session (`$debut` et `$nb_req`) mémorisent les informations sur le serveur.

```
// démarrage du mécanisme des sessions
session_start() ;

// déclaration des variables de session
    session_register("nb_req") ;
    session_register("debut") ;
    // incrémentation du nombre de requêtes
    $nb_req++ ;
    // inintialisation de l'heure de début de sessions
    if ( empty($debut) )
      $debut = time() ;
    // calcul du temps écoulé
    $temps = time() - $debut ;

function affichage_bas($titre,$h,$jour,$mois,$annee,$resultats,$type,$temps,$nb_req)
{
  $temps_str = ($temps % 60)."s" ;
  if ( $temps >= 60 )
    {
      $temps /= 60 ;
      $temps_str = ($temps % 60)."mn".$temps_str ;
      if ( $temps >= 24 )
        {
          $temps /= 24 ;
          $temps_str = $temps."h".$temps_str ;
        }
    }
  foreach ( file("modeles/bas15.html") as $ligne )
 {
      /* … */
      $ligne = str_replace("__TEMPS__",$temps_str,$ligne) ;
      $ligne = str_replace("__NB_REQ__",$nb_req,$ligne) ;
      echo $ligne ;
    }
}
```

Par défaut, les packages binaires livrés avec la distribution RedHat 7.1 n'ont pas le support de la ré-écriture transparente du code HTML pour propager les identificateurs de session comme des paramètres CGI. L'annexe A montre comment recompiler PHP en quelques commandes pour obtenir ce support.

Une fois les nouveaux packages installés, on s'aperçoit en regardant les sources produits de la ré-écriture effectuée. La fonctionnalité est alors active même pour les navigateurs refusant les cookies du site.



18. Créer des images dynamiques

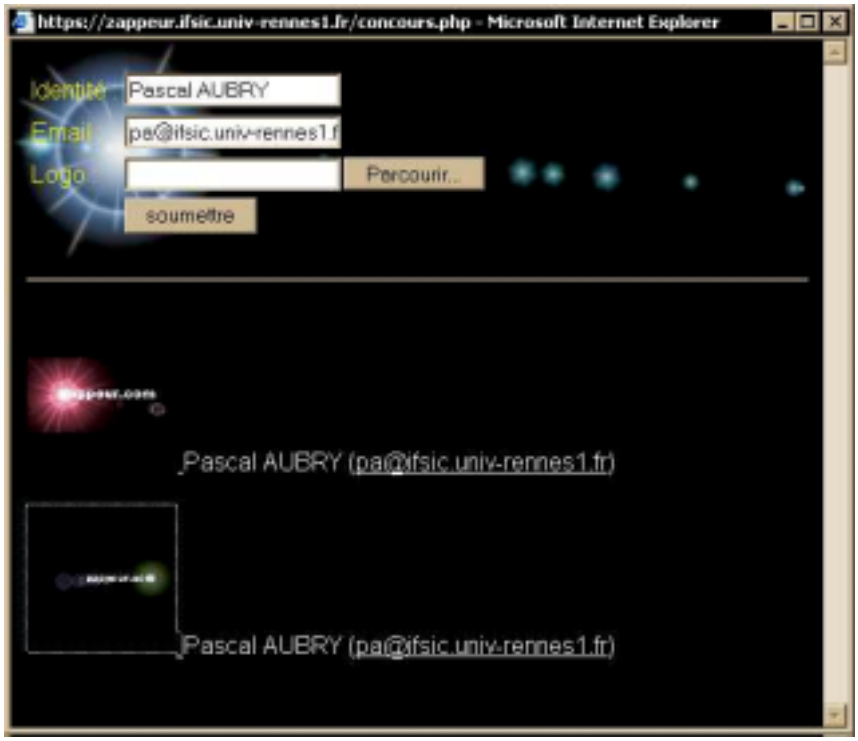
Le prétexte de l'exercice est un concours de logo organisé par zappeur.com. L'application offre un lien « concours », qui ouvre une interface qui permet aux utilisateurs :

- de soumettre un logo ;
- de connaître les logos déjà soumis.

La soumissions de nouveaux logos utilise le téléchargement de fichiers :

```php
// création d'un identificateur unique (pour sauvegarde)
$id = md5(uniqid(rand())) ;
// test pour vérifier que l'utilisateur a bien transmis un fichier
if ( $tmp_name == "none" )
  return ;
// test pour vérifier le format du fichier (PNG ou JPEG)
if ( !($src = @ImageCreateFromPNG("$tmp_name")) )
  if ( !($src = @ImageCreateFromJPEG("$tmp_name")) )
    return ;
// récupération de la taille de l'image (pour enregistrement)
$w = ImageSx($src) ;
$h = ImageSy($src) ;
ImageDestroy($src) ;
// copie du fichier téléchargé
if ( !move_uploaded_file($tmp_name,"logos/$id"))
  return ;
// enregistrement des informations du fichier
$f = fopen("logos/$id.info","w") ;
fputs($f,"$ident\n$email\n$w\n$h") ;
fclose($f) ;
```

Les logos déjà soumis sont réduits (à la taille 120x120) à la volée pour être affichés sur la page du concours :



```
$mini_size = 120 ;
// ouverture de l'image
if ( !($src = @ImageCreateFromPNG("logos/$name")) )
  if ( !($src = @ImageCreateFromJPEG("logos/$name")) )
    return ;
$dst = ImageCreate($mini_size,$mini_size) ;
// allocation dela couleur de fond
ImageColorAllocate($dst,0,0,0) ;
$src_sx = ImageSx($src) ; $src_sy = ImageSy($src) ;
// calcul des nouvelles tailles
if ( $src_sx > $src_sy )
  { $dst_sx = $mini_size ;
    $dst_sy = intval($mini_size * doubleval($src_sy) / doubleval($src_sx)) ;
    $dst_x = 0 ;
    $dst_y = ($mini_size - $dst_sy) / 2 ; }
else
  { $dst_sy = $mini_size ;
    $dst_sx = intval($mini_size * doubleval($src_sx) / doubleval($src_sy)) ;
    $dst_y = 0 ;
    $dst_x = ($mini_size - $dst_sx) / 2 ; }
// mise à la nouvealle taille
ImageCopyResized($dst,$src,$dst_x,$dst_y,0,0,$dst_sx,$dst_sy,$src_sx,$src_sy) ;
// sortie standard
header("Content-Type: image/png") ;
ImagePng($dst) ;
ImageDestroy($dst) ;
ImageDestroy($src) ;
```

Chaque logo soumis par les utilisateurs est présenté réduit, mais un lien sur chacun d'entre eux permet de les visualiser à taille réelle. On ajoute « à la volée » le nom du créateur de l'image.

```
// ouverture de l'image
if ( !($im = @ImageCreateFromPNG("logos/$name")) )
  if ( !($im = @ImageCreateFromJPEG("logos/$name")) )
    return
// allocation de la couleur d'écriture
$fg = ImageColorAllocate ($im, 255, 255, 255);
// lecture des infos
$info = file("logos/$name.info") ;
// affichage du nom du créateur de l'image
ImageTTFText($im,16,0,10,20,$fg,"comicbd.ttf",chop($info[0])) ;
header("Content-Type: image/jpeg") ;
ImageJpeg($im) ;
ImageDestroy($im) ;
```

19. Affichage des programmes au format XML

Bien que ce ne soit pas nécessaire, nous allons maintenant délivrer les programmes au format XML. Pour cela, il suffit, avant d'envoyer les données XML proprement dites, de spécifier dans l'entête HTTP de la réponse que le type MIME n'est pas celui par défaut (`text/html`), mais `text/xml`.


```
header("Content-Type: text/xml") ;
function affichage_xml($db,$type,$h,$jour,$mois,$annee) {
  header("Content-Type: text/xml") ;
  echo "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>
<zappeur>
  <info>
    <date>
      <year>$annee</year>
      <month>$mois</month>
      <day>$jour</day>
      <hour>$h</hour>
    </date>
    <type>$type</type>
  </info>
  <data>" ;
  $result = resultats_bruts($db,$type,$h,$jour,$mois,$annee) ;
  while ($row = $result->fetchRow(DB_FETCHMODE_ASSOC))
    {
      format_chiffre2($row["h"]) ;
      format_chiffre2($row["m"]) ;
      echo "
    <entry>
      <date>
        <year>".$row["annee"]."</year>
        <month>".$row["mois"]."</month>
        <day>".$row["jour"]."</day>
        <hour>".$row["h"]."</hour>
        <min>".$row["m"]."</min>
      </date>
      <canal>".$row["chaine"]."</canal>
      <title>".$row["libelle"]."</title>
      <type>".$row["type"]."</type>
      <detail>".str_replace("<br>","<br/>",$row["resume"])."</detail>
    </entry>" ;
    }
  $result->free() ;
  echo "
  </data>
</zappeur>" ;
}
```

Annexe : recompiler PHP pour obtenir le support transparent des sessions sans cookies (paramètres CGI)

Les packages binaires `php` et `mod_php` sont fournis avec certaines options de compilation qui ne sont pas forcément souhaitées ou suffisantes. Nous donnons ci-dessous la marche à suivre pour recompiler php avec d'autres options. Pour l'exemple, nous allons recompiler le package avec l'option `–enable-trans-sid`, qui autorise la ré-écriture du code HTML produit en transmettant l'identificateur de session en paramètres CGI.

Copier le package source de PHP dans `/usr/src/redhat/SRPMS` et l'installer :

```
[root@zappeur /root]# cd /usr/src/redhat/SRPMS
[root@zappeur SRPMS]# mount /dev/cdrom
[root@zappeur SRPMS]# cp /mnt/cdrom/SRPMS/php-4.0.4pl1-9.src.rpm .
[root@zappeur SRPMS]# rpm –ivh php-4.0.4pl1-9.src.rpm
Preparing...                ########################################################### [100%]
   1:php                    ########################################################### [100%]
 [root@zappeur SRPMS]# umount /dev/cdrom
 [root@zappeur SRPMS]# cd ../SPECS
[root@zappeur SPECS]# emacs rpm –ivh php-4.0.4pl1-9.src.rpm
```

Modifier `/usr/src/redhat/SPECS/php.spec` comme suit :

```
Compile() {
./configure \
        --prefix=%{prefix} \
        ...
        –-with-xml \
        --enable_trans_sid
make
}
```

Recompiler le package avec les nouvelles options :

```
[root@zappeur SPECS]# rpm –bb php.spec
... (quelques minutes ;-)
```

Les packages binaires produits se trouvent dans /usr/src/redhat/RPMS/i386. Il ne reste plus qu'à les installer :

```
[root@zappeur SPECS]# cd ../RPMS/i386
[root@zappeur i386]# rpm –ivh –-force php-4.0.4pl1-9.i386.rpm php-mysql-4.0.4pl1-9.i386.rpm
Preparing...                ########################################################### [100%]
   1:php                    ########################################################### [100%]
   2:php-mysql              ########################################################### [100%]
```

Il suffit ensuite de redémarrer le serveur Apache :

```
[root@zappeur i386]# /etc/init.d/httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd:                                            [  OK  ]
[root@zappeur i386]#
```

Notes