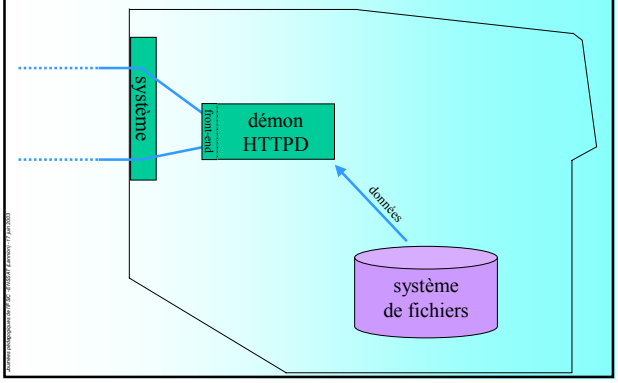
Exemple de transaction HTTP

```
% 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
```

Délivrer un document statique



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

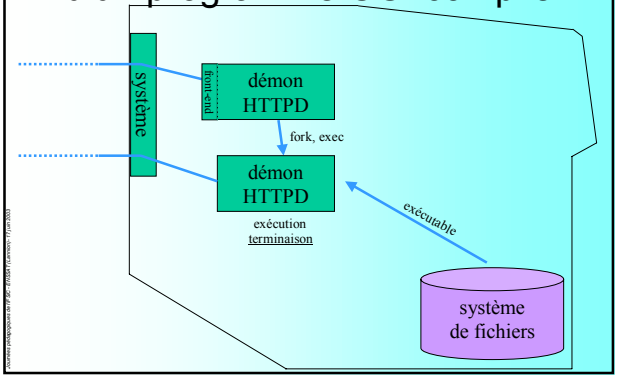
SSI, XSSI : le premier pas vers la dynamique

```
<!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>
```

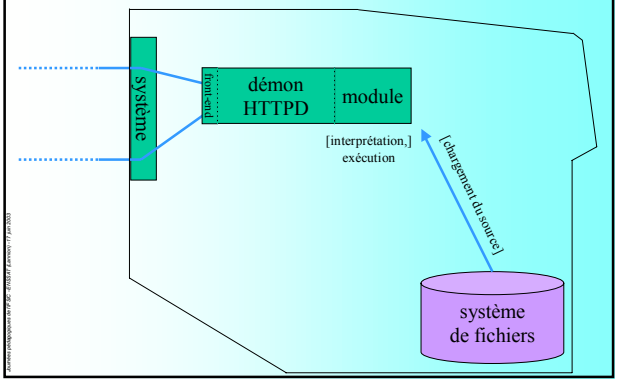
CGI : Common Gateway Interface

- 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



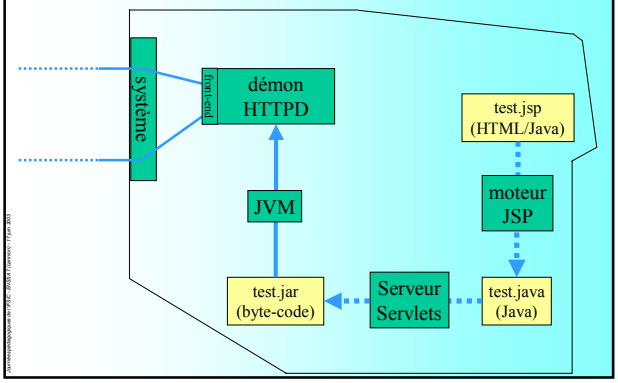
L'approche modulaire (PHP, Perl)



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.NET (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
  - Accès aux bases de données amélioré
  - Développement de web services

Une comparaison rapide…

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

…et subjective !

Un critère supplémentaire : la sécurité

- Tous les serveurs ont (eu) des trous de sécurité
  - Déni de service
  - Prise de contrôle
- Tous les navigateurs ont (eu) des trous de sécurité
  - Liés à Javascript essentiellement
  - Piratage de données (souvent)
  - Déni de service (parfois)
- On est au courant plus ou moins rapidement
  - Dans le monde du libre : correctifs rapides
  - Ailleurs : ?

Formulaires HTML

- Exemple :
  - `<form action="prog.php" method="get">`
    `<input type="submit" name="go">`
    `</form>`
- Action : URL cible
- Method : dans la pratique GET ou POST

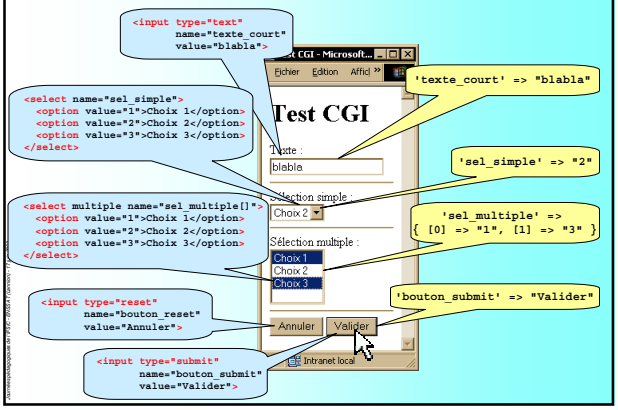
Entrées des formulaires HTML

- Boutons d'action
  - <input type="submit" name="nom" value="valeur">
  - <input type="reset" name="nom" value="valeur">
  - <input type="image" name="nom" src="source">
- Entrées textuelles
  - <input type="text" name="nom" value="valeur">
  - <input type="hidden" name="nom" value="valeur">
  - <input type="password" name="nom" value="valeur">
- Boites de saisie
  - <textarea name="nom" rows="r" cols="c">
    texte...
    </textarea>
- Listes à choix
  - <select [multiple] name="nom">
    <option value="valeur">texte</option>
    ...
    </select>
- Boutons radio
  - <input type="radio" name="nom" value="valeur" [selected] >
- Boutons à cocher
  - <input type="checkbox" name="nom" value="valeur" [checked] >

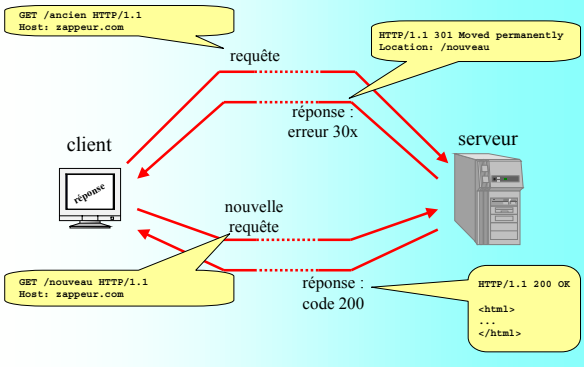
Exemple

```
<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>
```

Exemple



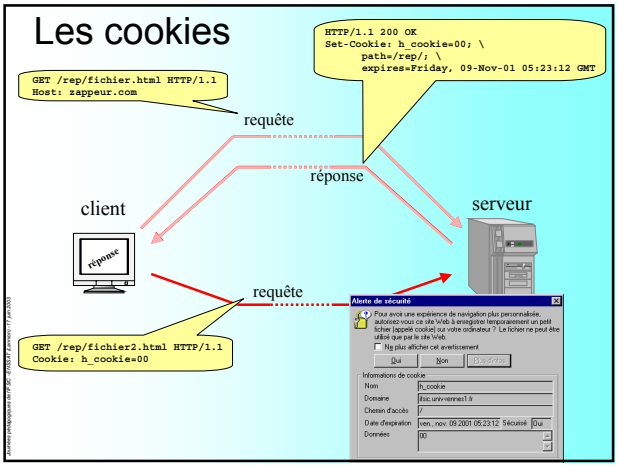
Redirection sous HTTP



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

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)

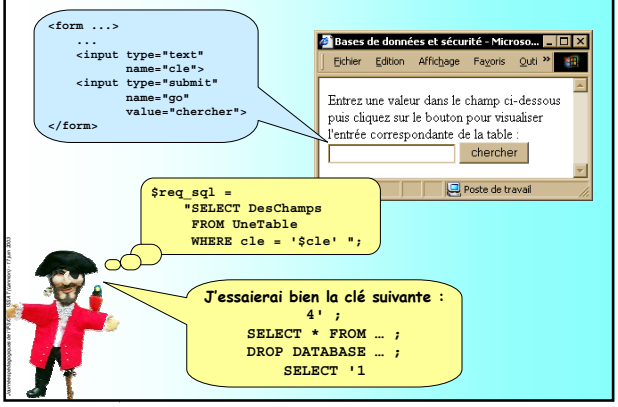
Transmission des IDs de session

- 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">`
  - `...`

Sécurité et CGI

- Tout ce qui provient de l'utilisateur doit être considéré comme suspect :
  - URL
    - Il ne suffit pas de protéger le point d'entrée d'une application
  - Paramètres CGI
    - L'utilisateur ne passe pas forcément par un formulaire de l'application
  - Cookies
    - Ils peuvent être « trafiqués »
  - Fichiers téléchargés
    - Leur contenu est a priori quelconque
  - Certificats X509
    - S'appuyer sur une autorité de confiance et une liste de révocation

Sécurité et CGI : exemple



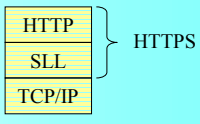
Sécurité et CGI

- Rappel : tout ce qui provient de l'utilisateur doit être considéré comme suspect
- Il existe des solutions

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

Le chiffrement de HTTP (HTTPS)

- SSL : Secured Socket Layout
- HTTPS : HTTP sur SSL
- Objectifs de HTTPS :
  - crypter les communications
  - identifier le serveur
    - le serveur auquel je suis connecté est-il le bon ?
  - Identifier le client
    - la personne qui se connecte au serveur est-elle la bonne ?
- Certificats (X509)



Chiffrement symétrique

- Chiffrement :
  f (message , clé) = message_chiffré

- Déchiffrement :
  f (message_chiffré,clé) = message

Chiffrement symétrique

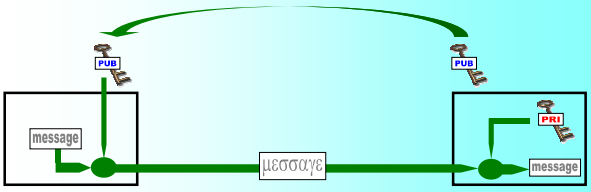
- Avantages :
  - Une seule clé pour chiffrer/déchiffrer
  - Algorithme de chiffrement léger
- Inconvénient :
  - Comment se passer la clé ?

Chiffrement asymétrique

- Chiffrement :
  f (message , clé_pub) = message_chiffré

- Déchiffrement :
  f (message_chiffré,clé_pri) = message

Communication chiffrée asymétrique

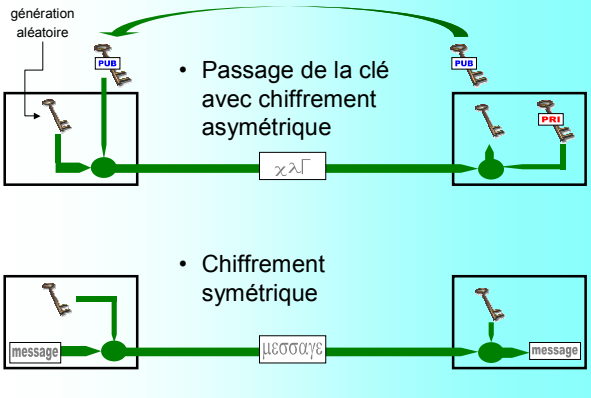


- Inconvénient :
  - L'algorithme de chiffrement/déchiffrement est lourd

La solution

- On utilise un chiffrement asymétrique pour se communiquer une clé symétrique
- On communique ensuite avec un chiffrement symétrique

La solution

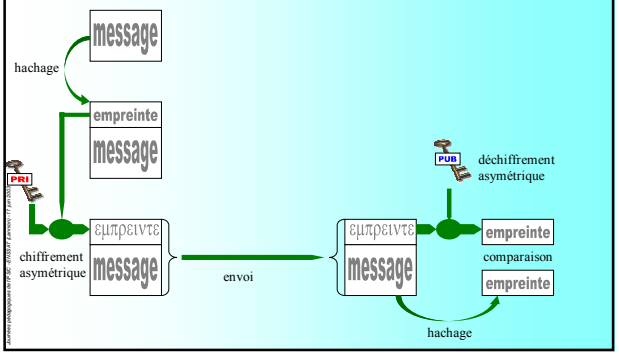


- Passage de la clé avec chiffrement asymétrique
- Chiffrement symétrique

La signature

- Être sûr que la personne/machine avec laquelle on communique est bien la bonne
- Être sûr que les données reçues sont bien celles qui ont été envoyées (intégrité)

La signature (fonctionnement)



L'authentification sur le web

- Quelle information d'authentification utilise-t-on ?
- Comment transporte-t-on les informations ?
- Qui vérifie les informations ?
- Par rapport à quoi les vérifie-t-on ?

Quelles informations d'authentification utiliser ?

- Un couple user/password
  - Méthode traditonnelle
  - Attention à son transport et à son utilisation !
  - Est-ce bien nécessaire ?
- Un jeton opaque
  - Rejouable ou non rejouable
  - Limité ou illimité dans le temps
  - Pour une gestion de session ou une authentification externalisée
- Un certificat X509

Comment transporte-t-on les informations d'authentification ?

- Sous HTTP
  - Par le protocole HTTP (royaumes)
    - Codage base64 => non sûr
  - Par paramètre CGI
    - Pas de chiffrement => non sûr
    - Log possible avec la méthode GET
  - Par cookie
    - Pas de chiffrement => non sûr
    - Piratage possible sur le client
  - Par certificat personnel X509
- Sous HTTPS
  - Tout est bon…

Qui vérifie les informations d'authentification ?

- Le démon HTTPD (le serveur web)
  - Ou un de ses modules
    - mod_ssl, mod_auth_ldap, mod_auth_dbm, mod_auth_mysql, …
  - Authentification gérée par l'administrateur
  - Peut être déléguée aux développeurs (.htaccess)
- L'application elle-même
  - L'application a en général accès aux informations d'authentification (royaumes HTTP, paramètres CGI, cookies, certificats X509)

Par rapport à quoi vérifie-t-on l'authentification ?

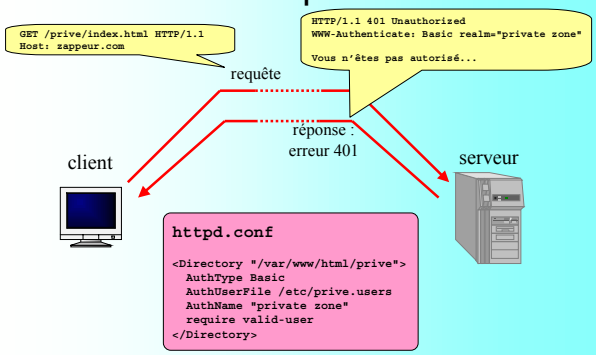
- Annuaire
  - LDAP (ex : mod_auth_ldap)
- Fichier
  - Séquentiel (ex : mod_auth_digest)
  - Format DBM (ex : mod_auth_dbm)
- Base de données
  - Ex : mod_auth_mysql
- Codage « en dur » dans l'application

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

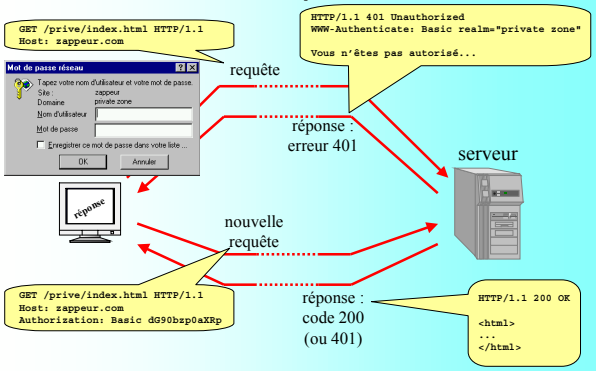
Exemple 1

- Mise à jour des pages web de l'IFSIC
  - Protocole : HTTP (en fait webDAV)
  - Informations : user/password
  - Transport : royaume HTTP
  - Vérification : serveur web
  - Référence : fichiers de mots de passe

Exemple 1



Exemple 1



Exemple 2

- Le Helpdesk de l'IFSIC
  - Protocole : HTTPS
  - Informations : certificats personnels
  - Transport : SSL
  - Vérification : application PHP
  - Référence : liste « en dur » des administrateurs

Besoin de SSO

- Un seul mot de passe : un objectif en général atteint
  - Par synchronisation des modes Unix et Windows
  - Par utilisation d'un annuaire unique (LDAP)
  - Risque accru au niveau sécurité
- Plusieurs authentifications
  - Une par système d'exploitation
  - Une par application
- Différents mécanismes
  - LDAP, PDC, NIS, certificats X509, …
  - Abstraction nécessaire
- D'autres problématiques encore
  - Aspects multi-établissements (inter-opérabilité)
  - Gestion d'autorisations

Les acteurs du marché

- MicroSoft Passport
  - Solution centralisée
- Sun Liberty Alliance
  - Plus un standard qu'une implémentation
  - De nombreux industriels impliqués
  - Solution répartie (relations de confiance)
- Yale CAS
  - La solution libre émergente
  - Adoptée par ESUP-Portail