Exemples de sessions

Exemple 1 : site marchand

Un site marchand propose à l'internaute de naviguer parmi les pages de descriptions de produits et de choisir des articles à mettre dans une « panier ».

- Chaque ajout d'article doit arriver dans le même panier.
- Quand l'internaute consulte le contenu de son panier, il doit y retrouver exactement les articles qu'il a choisi

Il y a donc nécessité pour le site de faire le **lien entre les différentes requêtes** d'un même internaute et de les différencier de celles d'un autre.
La suite de requêtes d'un internaute particulier va constituer une **session**.
Le serveur doit mémoriser des données (entre autres le contenu du panier) de chaque session ouverte.

Exemple de sessions

Exemple 2 : authentification

Un site n'est accessible qu'à certains utilisateurs, identifiés par nom et mot de passe (par exemple). Si on ne veut pas demander l'authentification à chaque page consultée, il faut créer une session.

- soit on lui redemande de s'authentifier à chaque page consultée (**c'est évidemment inacceptable**)
- ou alors le serveur doit pouvoir faire un lien entre ses différentes consultations de page du même internaute (« reconnaître l'internaute »)

Le client peut mettre fin à la session (« déconnexion ») et/ou celle-ci peut être annulée par le serveur après un certain temps d'inactivité.

Qu'est une session ?

Une **session est une suite de requêtes** qui

- concernent le même utilisateur connecté sur un même poste client, un même navigateur.
- sont liées entre elles (concernent le même sujet, la réponse à donner à une requête dépend des requêtes précédentes).
- une session peut nécessiter la mémorisation sur le serveur de données ou paramètres qui lui sont propres.

Pas de sessions en HTTP

Le protocole HTTP (pur) ne permet pas d'établir de lien entre différentes requêtes issues d'un même utilisateur.

- Les requêtes arrivent séparément et successivement, il est impossible de savoir si une requête est reliée à une autre.
- Le numéro ip **n'est pas pertinent** pour identifier l'utilisateur.

Une très mauvaise idée...

L'utilisation du numéro IP ne permet pas de connaître l'utilisateur :

- Des requêtes de plusieurs utilisateurs différents peuvent provenir de la même machine cliente (machine multi-utilisateurs, serveur mandataires-proxy, passerelle avec translation d'adresse).
- Une même machine cliente peut avoir des noms et adresses IP qui changent dans le temps (allocation dynamique de num IP).

Identifiant de session.

Création et transmission d'un identifiant

Le serveur doit attribuer **un identifiant** unique pour chaque session

- cet identifiant est transmis par le serveur au client lors du démarrage de la session
- le client doit ensuite présenter cet identifiant à **chaque** requête pour être «reconnu»

NB : Par des méthodes de cryptographie, on peut créer des numéros sécurisés (le serveur peut vérifier que le numéro de session est correct et authentique : c'est bien lui qui l'a créé).

Comment transmettre l'identifiant ?

1 : dans l'URL

- Ajouter l'identifiant comme variable HTTP à **chaque** lien renvoyant vers le site. Un lien aura une structure du genre `http://monsite.com/bidule.html?session=765865`
- Inconvénient : **Tous** les liens figurant doivent être réécrits dynamiquement pour ajouter ce paramètre
- Autre inconvénient : ce numéro de session peut aisément être dupliqué côté client (si un autre utilisateur récupère une page, il obtient aussi le numéro de session).

Comment identifier les requêtes ?

2 : les cookies

Les **cookies** constituent un mécanisme permettant à un serveur de **mémoriser des données dans le navigateur** du client .

- Un cookie peut contenir une information de taille modeste (4Ko).
- Tout serveur peut placer des cookies (sauf refus de l'utilisateur). **Un cookie est associé au nom du serveur qui l'a créé.**
- Un serveur prend connaissance des cookies qu'il a lui-même déposé **et uniquement** de ceux-là (confidentialité).

Les cookies sont véhiculés par le protocole HTTP : lors d'une requête, le navigateur transmet au serveur tous les cookies qui le concernent. Le serveur, lui, peut demander la mémorisation d'un cookie dans l'en-tête d'une réponse.

Comment identifier les requêtes ?

2 : les cookies

À la création de la session, le serveur utilise un cookie pour mémoriser, côté client, l'identificateur de session.
À chaque requête, le client transmet le cookie au serveur qui sait donc à quelle session la rattacher.
Avantages :

- Le numéro de session ne figure plus dans le contenu des pages. Il est plus difficilement copiable.
- La génération des pages est facilitée.

Inconvénient : Le navigateur doit accepter les cookies.

Bibliothèque PHP

PHP propose des **fonctions pour gérer les sessions**.

Par défaut, les cookies sont utilisés pour mémoriser les identifiants.

Le programmeur ne doit

- ni générer lui-même l'identifiant de session
- ni s'occuper de la mémorisation du cookie

Ce qu'apporte l'API des sessions :

- l'ouverture de session et la génération d'identifiants.
- le test permettant de connaître la session en cours.
- la **sauvegarde et la restauration du contexte de session** (dans un tableau nommé `$_SESSION`)

NB : `$_SESSION` à utiliser pour mémoriser des données pas trop volumineuses. Les données volumineuses ou complexes devront être mémorisées dans une base de données.

Bibliothèque PHP : fonctions principales

`session_start()` teste si une session est en cours.

- si le client a transmis un identifiant de session valide, alors `session_start()` **restaure** le contexte de la session (tableau `$_SESSION`)
  → restaure ce tableau tel qu'il était lors de la précédente requête rattachée à la même session.
- si non, **crée** une nouvelle session. Le tableau `$_SESSION` est initialement vide.

L'appel à `session_start()` doit être exécuté avant de produire une donnée quelconque Il doit se situer avant la balise `<HTML>`

Bibliothèque PHP : fonctions principales

`session_name(string)` : attribue un nom à la session

(en remplacement du nom par défaut).
Facultatif, mais doit être utilisé avant `session_start`
NB : sans argument, `session_name()` renvoie seulement le nom de la session courante.

`session_destroy()`

Cette fonction efface le contexte de session.
Toutes les données concernant la session seront perdues.
L'identifiant de session n'est pas détruit.

Schéma d'un script avec gestion de session.

```
<?php session_start()
 // Si une session est en cours pour cette
 // requête le contenu du tableau $_SESSION
 // concernant la session est restauré
?>
<html> ......
 <?php
   // pendant toute l'exécution du script on
   // peut retrouver ou modifier les infos
   // relatives à la session en cours
   // dans le tableau $_SESSION
 ?>
</html>
<!--  En fin de script le contenu du tableau
  $_SESSION est recopié dans une mémoire
  permanente (fichier) pour pouvoir être
  restauré lors d'un exécution ultérieure. -->
```