Cookies et Sessions

Cyril Rabat

http://cyril-rabat.fr

Programmation Web - Php

2019-2020

Cours Info 303

Gestion des cookies et sessions en PHP

Version 30 septembre 2019

Table des matières

Le suivi de navigation

- Le protocole HTTP est sans état :
  ↪Pas de suivi entre les scripts
- Le suivi de navigation :
  ↪Conservation des données du client tout au long des échanges
- Deux solutions :
  - Transférer les données à chaque échange :
    ↪Formulaire avec champs cachés
    ↪*cookie*
  - Stocker les données sur le serveur et l'associer à une clé :
    ↪Session

Le champ caché

- Dans un formulaire, possibilité de créer un champ caché
- Champs lisible par l'utilisateur (afficher la source) :
  ↪Possibilité de modifier cette valeur sans contrôle (injection)
- Obligation de passer d'une page à une autre via un formulaire :
  ↪Que faire des liens ?
  ↪Difficile à utiliser pour des données complexes

Exemple de champ caché

```
<input type="hidden" name="login" value="toto"/>
```

Le *cookie*

- Élément du protocole HTTP
- Enregistrement des données du côté client :
  ↪Associations clé/valeur (chaînes de caractères)
- Stockage sous forme de fichiers sur le client (dépend du navigateur)
- Données envoyées dans chaque requête HTTP :
  ↪Nom et valeur du *cookie* spécifiés dans l'en-tête
- Information limitée au site courant :
  ↪Impossible de récupérer un *cookie* d'un autre site

Limitation des *cookies*

- Suivant le navigateur :
  ↪La taille et le nombre de *cookies* sont limités
- Toutes les données doivent être transférées à chaque requête
- Stockage sous forme de chaîne de caractères :
  - Sérialisation des données
  - Séparation éventuelle par un délimiteur
  - Réalisation d'un *parsing* en PHP

Remarque

Des langages tels que PHP facilitent heureusement la récupération des données

Créer un *cookie* en PHP

- Utilisation de la fonction `setcookie` :
  - `name` : le nom (unique) du *cookie*
  - `value` : la valeur du *cookie* (une chaîne)
  - `expire` : la date d'expiration du *cookie* (*timestamp* UNIX)
  - `path` : chemin sur le serveur, sur lequel le cookie sera disponible
  - `domain` : domaine où le *cookie* est disponible
  - `secure` : *cookie* envoyé que si la connexion est sécurisée
  - `httponly` : *cookie* accessible uniquement via HTTP classique (pas via *Javascript* ; dépend du navigateur)
- Rappel : le *cookie* est un élément du protocole HTTP, envoyé dans l'entête :
  ↪Appel à `setcookie` **avant** l'envoi de données

Récupérer ou modifier la valeur d'un *cookie*

- Variable superglobale `$_COOKIE` (tableau associatif)
  ↪Correspond aux *cookies* reçus (pas ceux créés dans le script)
- Pour récupérer la valeur d'un *cookie* : depuis le tableau
- Pour modifier la valeur d'un *cookie* :
  ↪Fonction `setcookie` !
- Pour détruire un *cookie* :
  ↪Spécifier une date de validité antérieure à la date actuelle

Exemple d'utilisation d'un *cookie* (1/3)

Script PHP `script1.php`

```
<?php
setcookie("moncookie", "valeur␣du␣cookie", time() + 3600);
?>
<html>
  <head>
    <title>Dépôt d'un cookie</title>
    <meta charset="UTF-8">
  </head>
  <body>
<?php
if(isset($_COOKIE["moncookie"]))
    echo "<p>␣Le␣cookie␣'moncookie'␣était␣déjà␣présent.␣</p>";
else
    echo "<p>␣Cookie␣mis␣en␣place.␣</p>";
?>
    <p> <a href="script1.php"> Allez vers la page 1 </a> </p>
    <p> <a href="script2.php"> Allez vers la page 2 </a> </p>
  </body>
</html>
```

Exemple d'utilisation d'un *cookie* (2/3)

Script PHP `script2.php`

```
<?php
if(!isset($_COOKIE["moncookie"]))
    header("Location:␣script1.php");

$valeur = $_COOKIE["moncookie"];
setcookie("moncookie", "", time() - 1);
?>
<html>
  <head>
    <title>Récupération d'un cookie</title>
  </head>
  <body>
    <p> Valeur du cookie : <?php echo $valeur; ?>. </p>
    <p> Le cookie est supprimé. </p>
    <p> <a href="script3.php"> Allez vers la page 3 </a> </p>
  </body>
</html>
```

Exemple d'utilisation d'un *cookie* (3/3)

Script PHP `script3.php`

```
<html>
  <head>
    <title>Vérification de l'existence d'un cookie</title>
  </head>
  <body>
<?php
if(isset($_COOKIE["moncookie"]))
    echo "<p> Valeur du cookie : ".$COOKIE["moncookie"].". </p>";
else
    echo "<p> Pas de cookie trouvé. </p>";
?>
    <p> <a href="script1.php"> Allez vers la page 1 </a> </p>
    <p> <a href="script2.php"> Allez vers la page 2 </a> </p>
  </body>
</html>
```

Les *cookies* de session

- Un *cookie* de session :
  - Conservé tant que l'utilisateur est sur le site
  - Supprimé automatiquement (quand l'utilisateur ferme son navigateur)
- Utilisation de `setcookie` sans préciser la date de validité
- Certains naviguateurs peuvent avoir des comportements spécifiques :
  ↪*Chrome* continue de fonctionner même si tous les onglets sont fermés !

Exemple de *cookie* de session

Script PHP `session.php`

```php
<?php
$compteur = 1;
if(!isset($_COOKIE["compteur"]))
    setcookie("compteur", 1);
else {
    $compteur = intval($_COOKIE["compteur"]) + 1;
    setcookie("compteur", $compteur);
}
?>
<html>
  <head>
    <title>Cookies de session</title>
  </head>
  <body>
    <p> La valeur du compteur est <?php echo $compteur; ?>. </p>
    <p> <a href="session.php"> Recharger </a> </p>
  </body>
</html>
```

Exercices

Exercice 4

- Écrivez plusieurs scripts PHP
  ↪`script1.php`, `script2.php`, *etc.*
- À l'aide d'un *cookie*, réalisez l'historique des visites des scripts
  ↪Limitez l'historique aux cinq derniers scripts visités
- Sur chaque script :
  - Affichez l'historique des visites (avec les liens)
  - Affichez les liens vers les autres scripts
- Indications :
  - Créez une classe `Historique`
  - Vous pouvez utiliser `implode` et `explode`
  - `array_unshift` : ajoute un élément en début de tableau
  - `array_slice` : extrait une partie d'un tableau

Table des matières

La session

- Génération d'une clé sur le serveur, envoyée au client
- À chaque échange (HTTP), envoi uniquement de la clé par le client :
  ↪Le client est donc "*reconnu*" par le serveur
- Sur le serveur, une session est créée :
  - La clé est associée à la session de ce client
  - Possibilité d'associer un jeu de données **stocké sur le serveur**
    ↪Sérialisé/dé-sérialisé automatiquement
  - Tout type de données possible (pour les objets, attention aux inclusions)
- Variable superglobale : `$_SESSION`
  ↪Tableau associatif contenant toutes les données

Transmission de la clé de session

- Deux (principales) solutions :
  - Utilisation d'un *cookie* de session
  - Spécifier la clé dans l'URL ou par la méthode POST
- La configuration par défaut de PHP supporte les deux :
  - Si les *cookies* sont acceptés, création d'un *cookie* de session
  - Sinon, modification automatique des URL (sauf externes !)
- La création/récupération de la session est automatique :
  - Si l'identifiant est récupéré, la session précédente est restaurée
  - Sinon, une nouvelle session est créée

Méthodes associées aux sessions en PHP

- `session_start()` : démarre une nouvelle session
- `session_destroy()` : détruit la session en cours
- Pour récupérer des informations sur la session en cours :
  - `session_name()` : nom de la session
  - `session_id()` : identifiant de la session
    ↪Ou constante `SID` (définie si la session a débuté)

Utilisation d'une session

Exemple d'utilisation d'une session

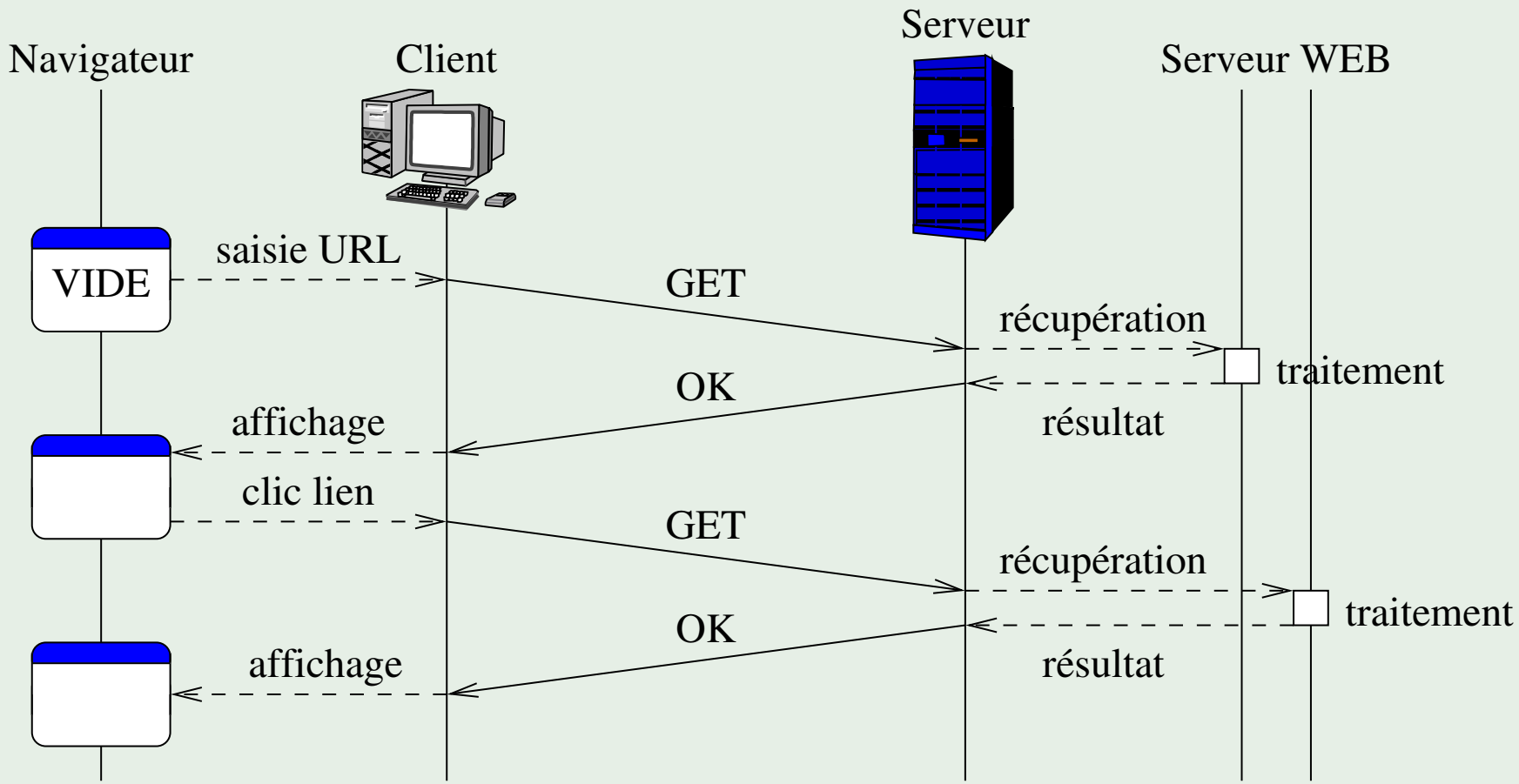
```php
<?php
session_start();

if(!isset($_SESSION['compteur']))
  $_SESSION['compteur'] = 1;
else
  $_SESSION['compteur']++;

echo "Valeur du compteur : ".$_SESSION['compteur']."<br/>";
?>
```

Différences entre sessions et *cookies*

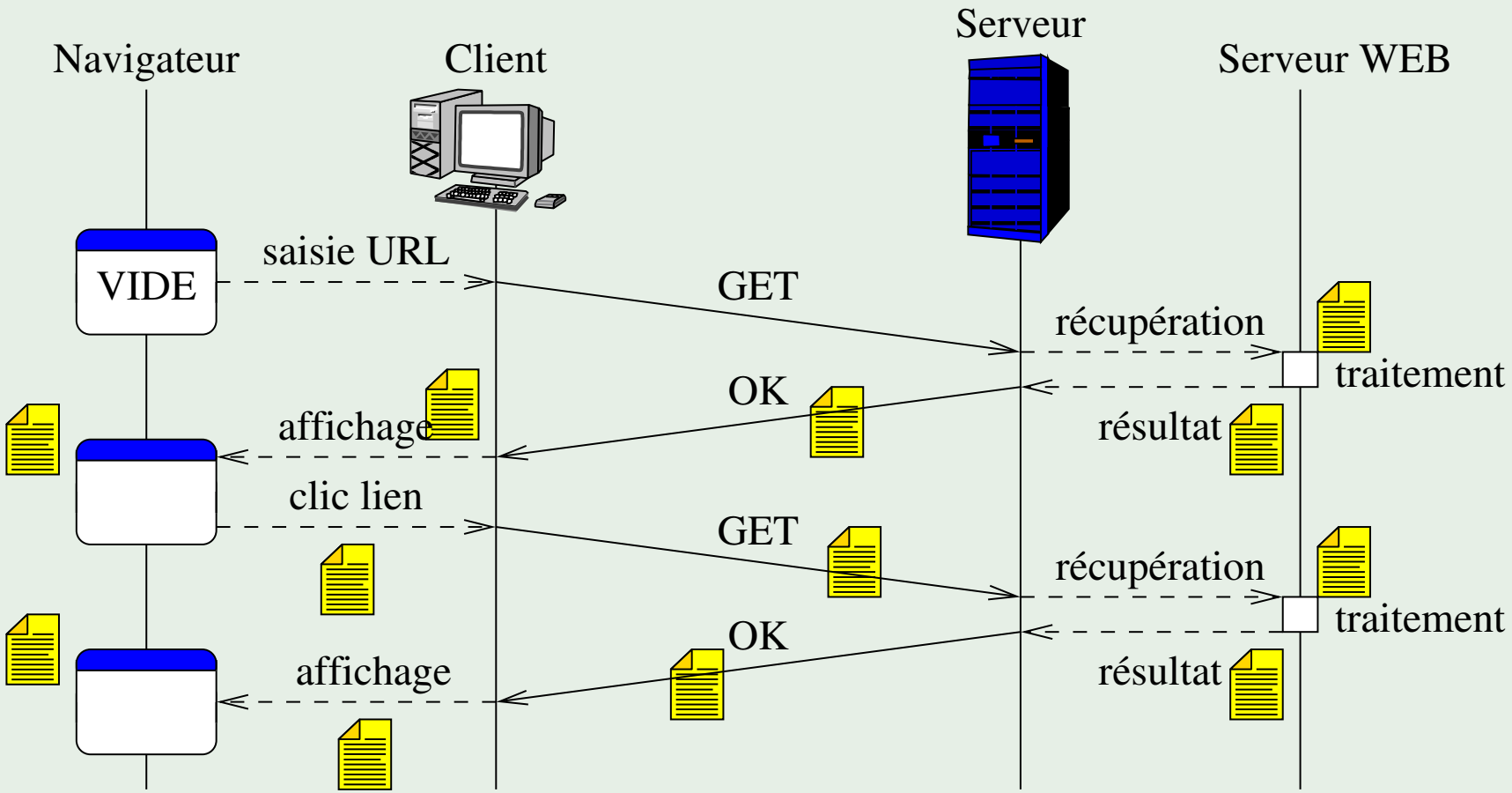
Illustration



Série de requêtes/réponses HTTP classiques

Différences entre sessions et *cookies*

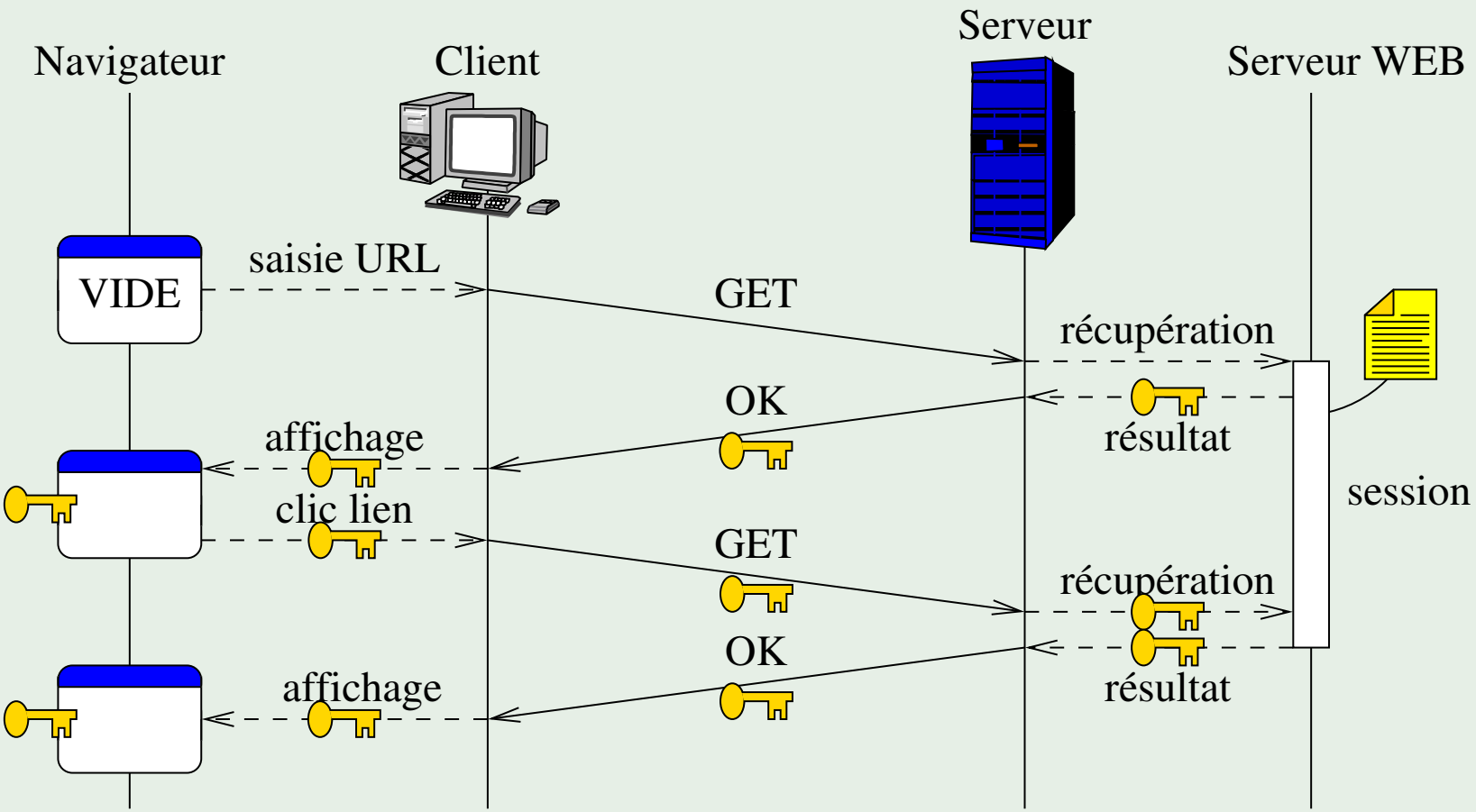
Illustration



Utilisation d'un `cookie`

Différences entre sessions et *cookies*

Illustration



Utilisation d'une session

Exercices

Exercice 5 (1/2)

- Écrivez une classe `Utilisateur` :
  - Méthode `afficherForm` :
    ↪Affiche un formulaire pour la saisie des identifiants
    ↪Ou un formulaire de déconnexion
  - Méthode `recupererForm` :
    ↪Récupération des données depuis un formulaire
- Lorsque l'utilisateur est récupéré depuis le formulaire :
  ↪Ajout en session d'un objet `Utilisateur`
- Ajoutez les méthodes :
  - `estLogue` : vérifie si un utilisateur est actuellement logué
  - `delogue` : "délogue" l'utilisateur courant

Exercice 5 (2/2)

- Créez plusieurs scripts PHP/HTML :
  - `index.php` :
    - Affichage du formulaire de connexion ou déconnexion
    - Lien vers les autres scripts si utilisateur connecté
  - `script1.php` et `script2.php` :
    - Affichage quelconque
    - Redirection vers `index.php` si non connecté
- Indications :
  - Redirection vers un script (avant affichage) :
    `↪header('Location: script.php');`
  - Appelez la méthode `recupererForm` en début de chaque script