PHP :

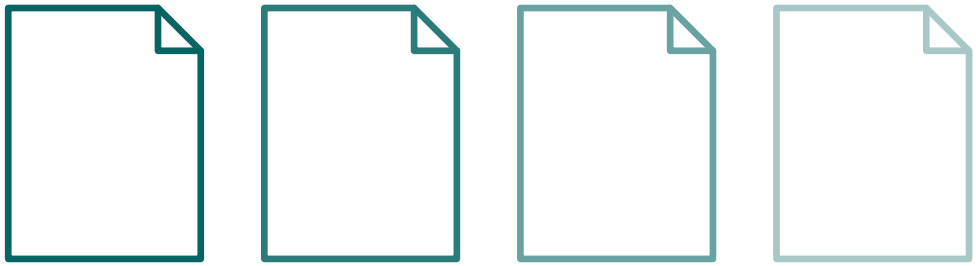
Fonctionnement des **sessions**

Protéger une section d'un site - établir et contrôler une session

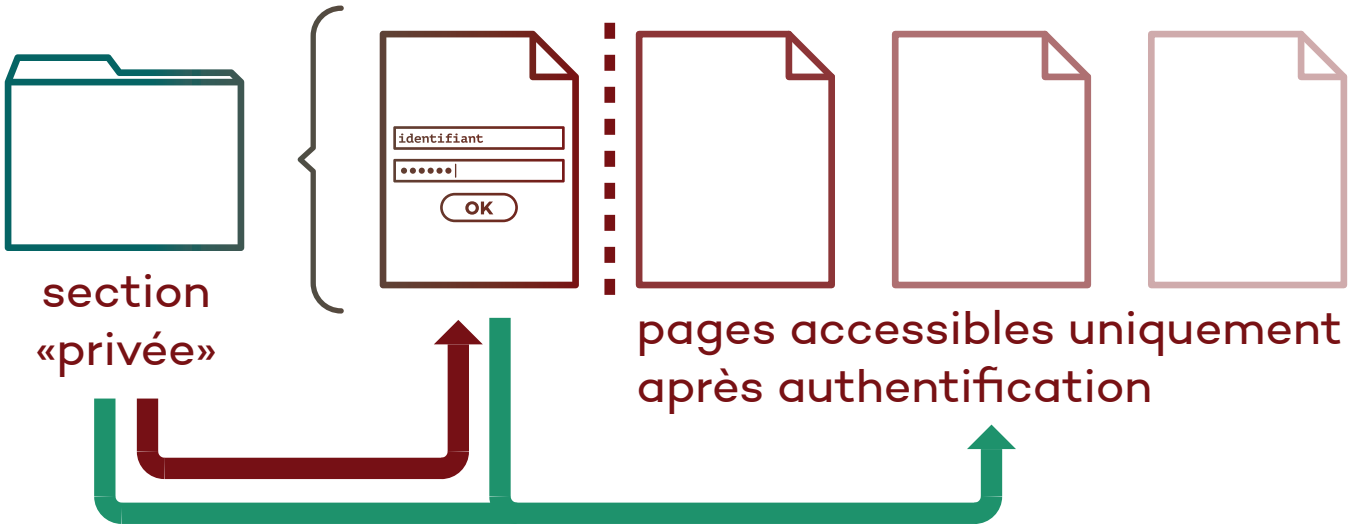
Demander au visiteur / utilisateur de s'identifier pour accéder à certaines sections du site ou de l'application est un besoin fréquent. On rencontre le cas, par exemple, dès qu'il est question d'une partie «back-office», d'un CMS, ou d'un compte pour un service quelconque : un réseau social, une boutique en ligne, un webmail, etc.

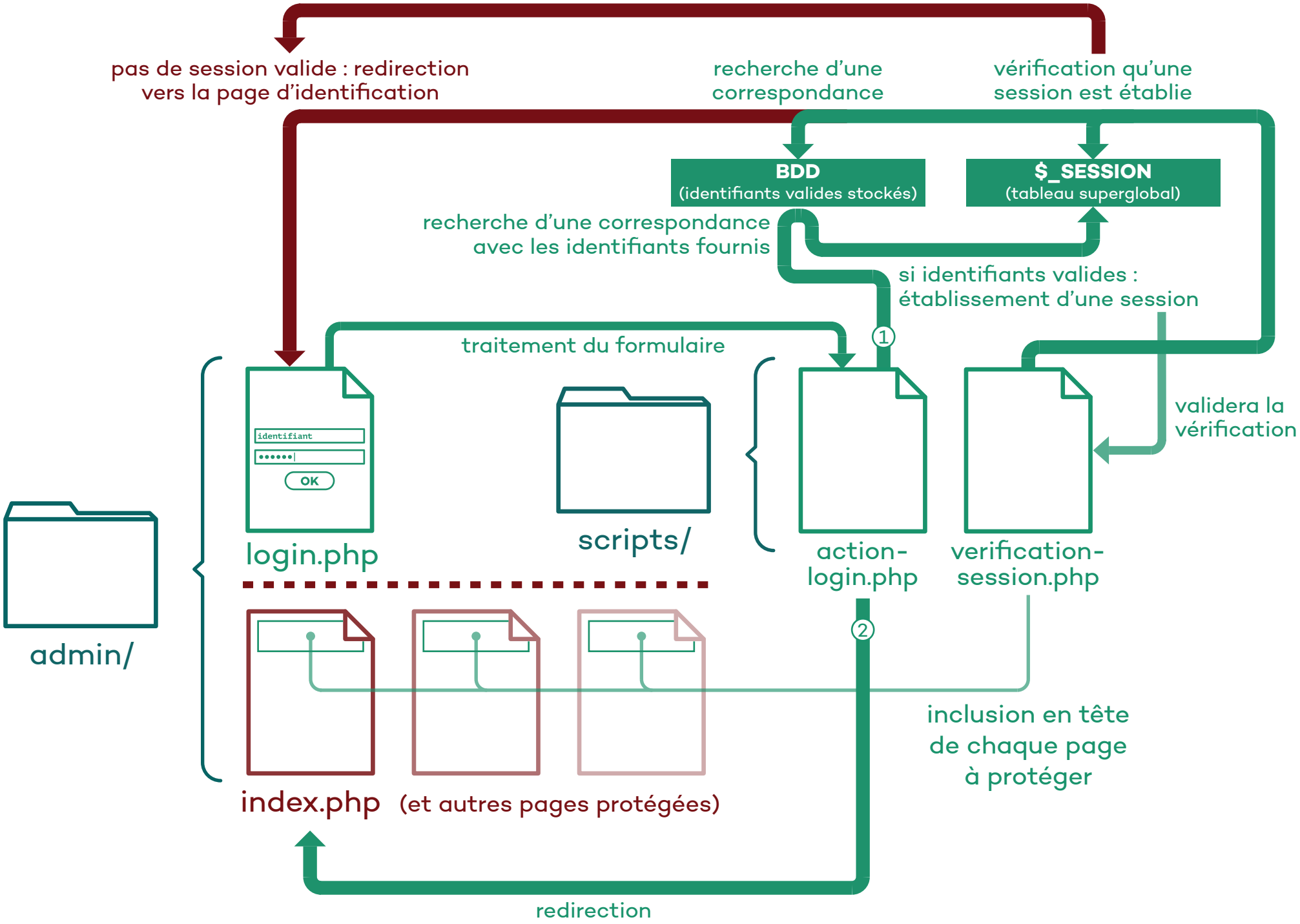
Il s'agira en fait de permettre, après renseignement d'identifiants dans un formulaire, l'établissement d'une session. Protéger une page consistera ensuite à y vérifier l'existence et la validité de cette session, pour que selon le cas le visiteur puisse rester sur la page ou soit immédiatement redirigé autre part.

En général, une section «admin» ou «mon compte» correspondra à un dossier / répertoire, contenant son propre site ou application et dont toutes les pages seront sujettes au contrôle d'une session. Seule la page d'identification (formulaire) sera «publiquement» accessible ; chaque visiteur non identifié pourra également y aboutir lorsque redirigé depuis une page protégée.



pages accessibles à tout visiteur




pas de session valide : redirection
vers la page d'identification
recherche d'une
correspondance
vérification qu'une
session est établie
BDD
(identifiants valides stockés)
$_SESSION
(tableau superglobal)
recherche d'une correspondance
avec les identifiants fournis
si identifiants valides :
établissement d'une session
traitement du formulaire
1
identifiant
OK
login.php
scripts/
action-
login.php
verification-
session.php
validera la
vérification
admin/
2
inclusion en tête
de chaque page
à protéger
index.php
(et autres pages protégées)
redirection

Côté BDD - stockage des identifiants valides

Une table permet de stocker les différents identifiants valides possibles, par exemple sous forme de couples nom d'utilisateur / mot de passe. Chaque ligne représentera un «compte» enregistré, et la vérification à effectuer pour authentifier un utilisateur consistera à chercher dans la table un compte dont à la fois le nom et le mot de passe correspondent à ceux qu'il aura fournis.

table **sessions**

id	login	motdepasse
1	administrateur01	*A4B61573190387 24E356089...

Les mots de passe peuvent être cryptés lors de leur ajout dans la table, et au final ne jamais figurer en clair nulle-part. La recherche de correspondance pourra se faire sans décrypter aucune donnée de la table ; il faudra simplement s'arranger pour crypter de la même manière la chaine à comparer, lors de l'envoi de la requête.

La fonction SQL PASSWORD() peut être utilisée pour le cryptage. Un résultat typique est observable dans la colonne motdepasse ci-dessus, où la valeur insérée correspond en fait à PASSWORD("1234"). Cette fonction n'est pas réversible (retrouver «1234» à partir de ce résultat de PASSWORD() est techniquement impossible).

```php
<?php

$login = "nom_utilisateur_fourni";
$mdp = "mot_de_passe_fourni";

$requete = "SELECT * FROM `sessions`
            WHERE `login` = '$login'
            AND `motdepasse` = PASSWORD('$mdp')";

$reponse = $connexion_sql->query($requete);

if (mysqli_num_rows($reponse) > 0) {

   // Si la réponse contient au moins une ligne,
   // une correspondance a été trouvée :
   // les identifiants fournis sont valides.

}

?>
```

Note : une fonction de cryptage personnalisée en PHP peut être employée à la place de PASSWORD(). Il faudra s'assurer de son aspect unidirectionnel, et utiliser la même fonction à l'ajout d'une ligne (création de compte) et lors de la vérification de session.

$_SESSION (tableau superglobal)

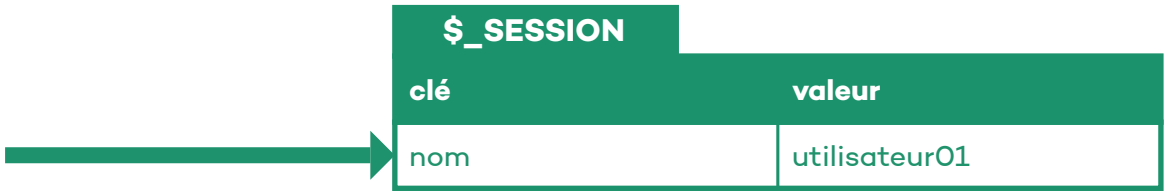
De la même manière que $_POST et $_GET, $_SESSION est un tableau associatif (chaque élément y possède une clé et une valeur), accessible selon certaines conditions depuis toute page PHP.

On y stockera diverses données relatives à une session utilisateur, comme par exemple son identifiant. Cela suffira à «établir» une session ; on pourra ensuite vérifier, depuis chaque page concernée, la présence (et éventuellement la validité) de cette valeur «identifiant» dans le tableau $_SESSION.

Pour pouvoir accéder à $_SESSION, il faut commencer par exécuter la fonction session_start(). Cela initialisera le tableau s'il n'existe pas encore, et permettra d'y maintenir les données d'une page à l'autre lorsqu'il est renseigné.

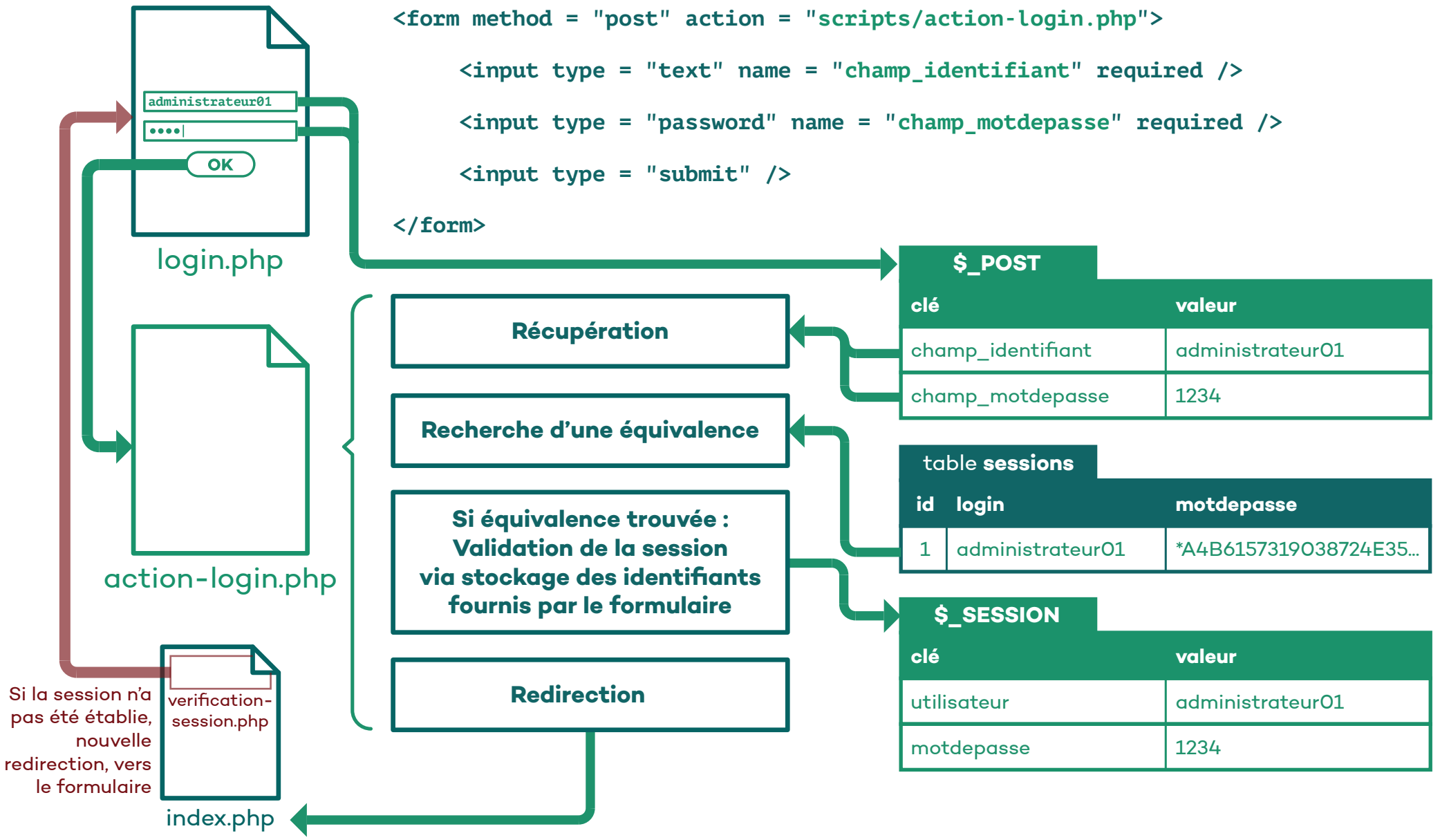
Important : session_start() ne peut pas être appelée une fois que du contenu a été affiché sur la page. Comme header(), cette fonction intervient sur l'en-tête HTTP, or dès que du texte doit être affiché, l'en-tête est envoyé et ne peut plus être modifié.

```php
<?php // Établir une session :
session_start();
$_SESSION["nom"] = "Utilisateur01";
?>
```




```php
<?php // Vérification (depuis une autre page) :
session_start();
if (isset($_SESSION["nom"])) {
   echo "Session établie pour l'utilisateur ".$_SESSION["nom"];
}
?>
```

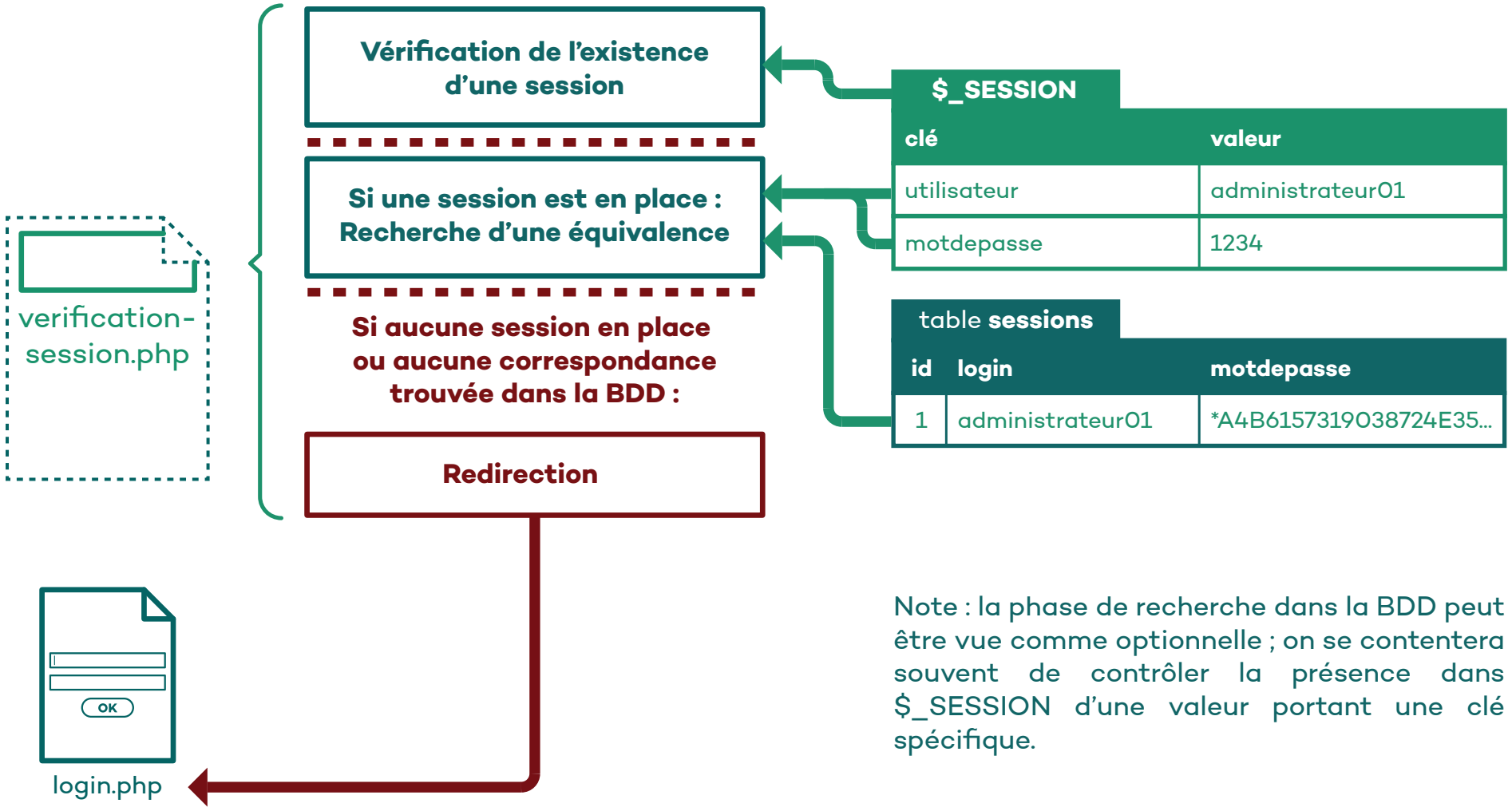
Authentification - formulaire HTML & script PHP / SQL



action-login.php - en détail :

```php
<?php

session_start();

if (isset( $_POST["champ_identifiant"] , $_POST["champ_motdepasse"] )) {

    $login = $_POST["champ_identifiant"];
    $mdp = $_POST["champ_motdepasse"];

    $connexion = mysqli_connect("localhost", "root", "root", "nom_de_la_BDD");

    $requete = "SELECT * FROM sessions WHERE `login` = '$login' AND `motdepasse` = PASSWORD('$mdp')";
    $reponse = $connexion->query($requete);

    if (mysqli_num_rows($reponse) > 0) {

        $_SESSION["utilisateur"] = $login;
        $_SESSION["motdepasse"] = $mdp;

    }

    mysqli_close($connexion);

}

header("location:../index.php");

?>
```

verification-session.php - «chasser» de la page un utilisateur non authentifié



Note : la phase de recherche dans la BDD peut être vue comme optionnelle ; on se contentera souvent de contrôler la présence dans $_SESSION d'une valeur portant une clé spécifique.

verification-session.php - en détail :

```php
<?php
session_start();

$authentification_ok = true;

if (isset($_SESSION["utilisateur"])) {

    $login = $_SESSION["utilisateur"];
    $mdp = $_SESSION["motdepasse"];

    $connexion = mysqli_connect("localhost", "root", "root", "nom_de_la_BDD");

    $requete = "SELECT * FROM `sessions` WHERE `login` = '$login' AND `motdepasse` = PASSWORD('$mdp')";
    $reponse = $connexion->query($requete);

    if (mysqli_num_rows($reponse) == 0) {
        $authentification_ok = false;
    }

    mysqli_close($connexion);

} else {
    $authentification_ok = false;
}

if (!$authentification_ok) {
    header("location:formulaire-login.php");
}

?>
```

La finalité de ce script est de rediriger, si besoin, l'utilisateur. Comme plusieurs facteurs peuvent rendre cette redirection nécessaire, mais qu'il est plus propre de n'appeler la redirection qu'une seule fois et en-dehors des divers test, on utilise ici une variable booléenne $authentification_ok.

Initialement établie comme vraie, elle sera vérifiée en fin de script, après avoir éventuellement été modifiée entre temps dans le cadre de tests (si aucune session en place / si aucune correspondance trouvée dans la BDD).

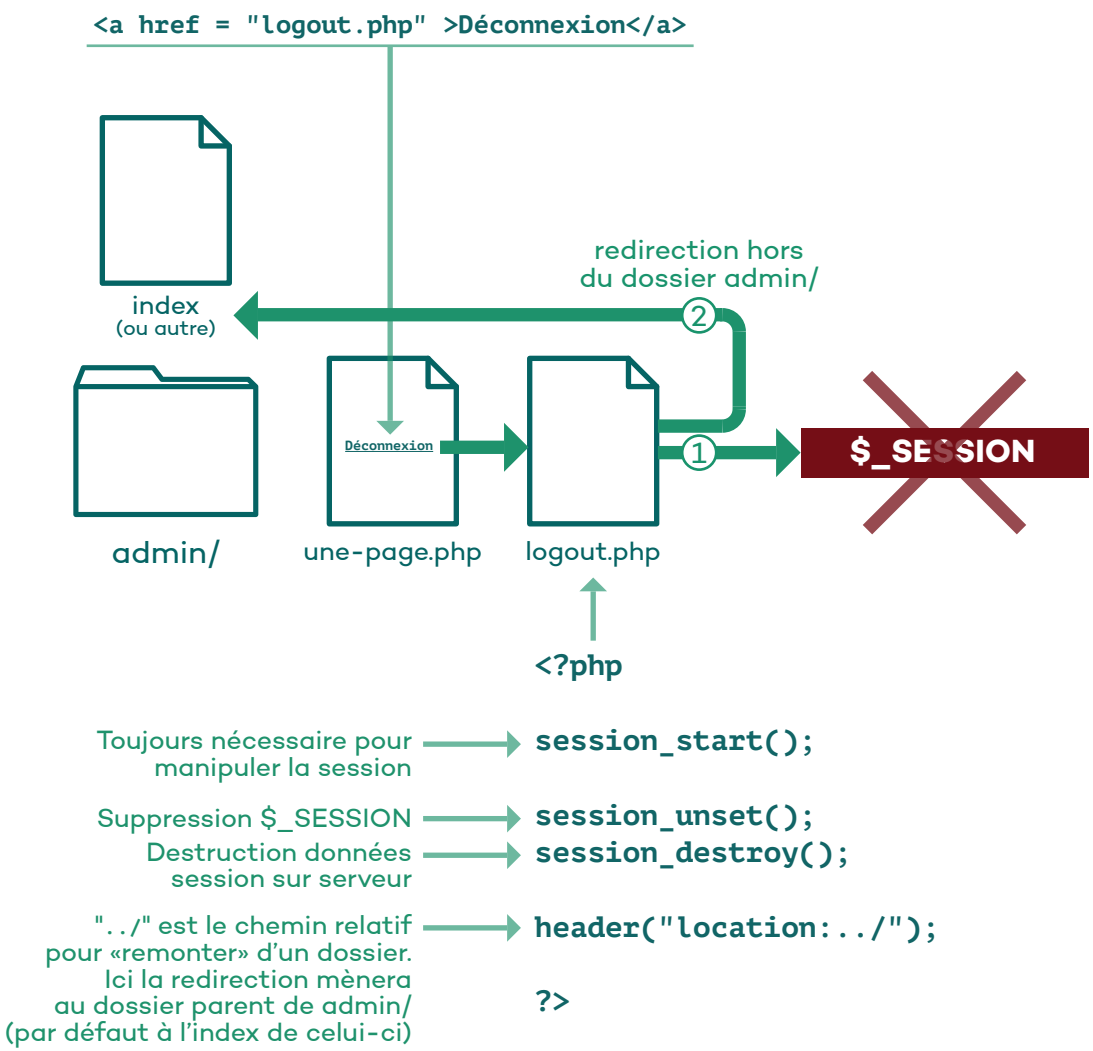
Si à ce moment la variable indique que l'authentification n'est plus validée, la redirection doit avoir lieu.

Fermeture d'une session

Pour des raisons de sécurité, une session possède une durée de vie limitée (définie dans les fichiers de configuration du serveur - par défaut une vingtaine de minutes dans la plupart des cas). Après ce délai, si la session n'a pas été «rafraichie» entre temps via un nouvel appel à session_start(), elle expirera, et le tableau $_SESSION sera détruit. Selon le navigateur utilisé pour établir la session, ce sera également le cas lorsque l'utilisateur quitte le navigateur.

Chaque appel à session_start() remet à zéro ce délai, ré-activant la session à chaque nouvelle page PHP concernée que visite l'utilisateur.

Une session peut également être volontairement détruite. Il s'agit d'ailleurs d'un point important : l'utilisateur doit pouvoir refermer l'accès à son compte dès qu'il le souhaite. On peut le mener directement à un script PHP qui s'en chargera, avant de rediriger l'utilisateur hors de la section privée.



Annexe - Redirection vers "../" - rendre un dossier non listable

Dans la structure d'un site ou d'une application, certains dossiers servent uniquement à ranger des ressources (images, scripts, etc) et ne sont pas à considérer comme des sections visitables par un utilisateur : aucune véritable page n'y figure.

En l'absence d'un fichier index dans un dossier, ce qu'affiche par défaut un navigateur lorsque l'utilisateur s'y rend est une liste, sous forme de liens, de ses contenus.

On peut automatiquement «jeter dehors» d'un tel dossier les utilisateurs tentant de s'y rendre, en y plaçant un index.php très simple, constitué uniquement du bloc suivant :

```
<?php

header("location:../");

?>
```

Notes :

• Il est possible de remonter plus d'un dossier à la fois. Exemple : **`header("location:../../../")`** ramènera trois dossiers plus «haut» (dans l'arborescence).

• Le retour peut aussi être utilisé dans des chemins plus spécifiques. Exemple :
`header("location:../section/page-specifique.html")`