DANTENBANKHANDLING

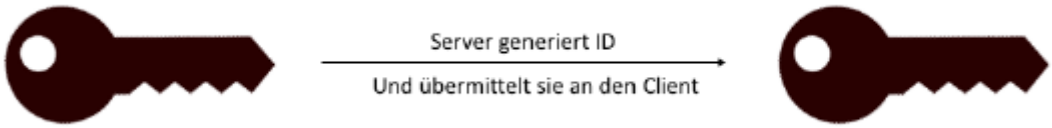
Kompetenznachweis

Adriana Arteaga, Nuria Anaya
Adreas Nydegger, Kevin Schleuniger

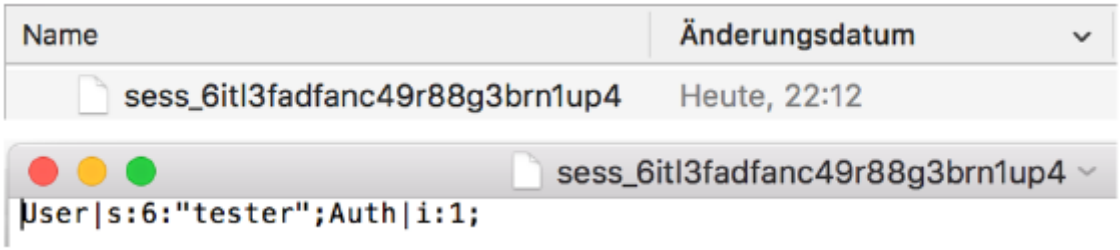
Inhalt

Session

Eine Session bedeutet Sitzung, es beschriebt eine stehende Verbindung zwischen Client und Server. Eine Session fängt deshalb mit dem Login an und endete mit dem Logout. Die Session-IDs verwenden zustandlose Protokolle (HTTP/HTTPS) um mehrere zusammengehörenden Anfragen eines Benutzers zu erkennen. Man soll die Wahrscheinlichkeit, zufällig auf eine gültige Session-ID zu stoßen, verschwindend klein sein, indem man einen zufälligen Wert in einem großen Wertebereich benutzt.



Wie können sich nun also den Client und den Server erkennen? Die Session-ID muss zwischen Client und Server ausgetauscht werden. Dies geschieht entweder als Cookie im http-Header, als Query-Parameter in der URL -> GET oder als Datenteil eines Formulars-> POST.
Die Session-Date werden in einem Verzeichnis auf dem Webserver gespeichert z.B. C:/XAMPP/tmp



Beispiel

Code	Erklärung
session_start();	Die Funktion muss in jedem PHP-Script aufgerufen werden, welches auf die Session zugreifen will.
$_SESSION	Ist ein superglobales, assoziatives Array, steht zur Verfügung wen die Funktion session_start() aufgerufen wurde
$_SESSION[‚username‘] = $username; $_SESSION[‚loggedin‘] = $true;	Schreiben in die Session

If(isset($_SESSION[‚loggedin‘])){
Echo „Hallo“.$_SESSION[‚username‘];
}
	Prüfen, ob die Session-Variable besteht

$_SESSION = array();	Die Session besteht weiterhin, lediglich die gespeicherten Informationen sind gelöscht
Session_destroy();	Nun besteht keine Verbindung mehr zwischen Client und Server

Stored Procedures

Stored Procedures (gespeicherte Prozedur) ist eine Reihe von SQL-Anweisungen mit einem zugewiesenen Namen, die in einem relationalen Datenbankverwaltungssystem als Gruppe gespeichert wird, sodass sie von mehreren Programmen wiederverwendet und gemeinsam genutzt werden können.
Gespeicherte Prozeduren können auf Daten in einer Datenbank zugreifen oder diese manipulieren, sind jedoch nicht an eine bestimmte Datenbank oder ein bestimmtes Objekt gebunden, was eine Reihe von Vorteilen bietet.

Vorteile

- Eine gespeicherte Prozedur bietet eine wichtige Sicherheitsebene zwischen Benutzeroberfläche und der Datenbank. Es unterstützt die Sicherheit durch Datenzugriffskontrollen, da Endbenutzer zwar Daten eingeben oder ändern, aber keine Prozeduren schrieben können.
- Eine gespeicherte Prozedur bewahrt die Datenintegrität, da Informationen auf konsistente Weise eingegeben werden. Dies verbessert die Produktivität, da Anweisungen in einer gespeicherten Prozedur nur einmal geschrieben werden müssen.
- Gespeicherte Prozeduren bieten Vorteile gegenüber dem Einbetten von Abfragen in eine grafische Benutzeroberfläche, da gespeicherte Prozeduren modular aufgebaut sind, ist die Problembehandlung einfacher, wenn in einer Anwendung ein Problem auftritt. Prozeduren können ebenfalls angepasst werden, sodass der GUI-Quellcode nicht mehr geändert werden muss
- Durch die Verwendung gespeicherter Prozeduren kann der Netzwerkverkehr zwischen Client und Server verringert werden, da die Befehle als einzelner Codestapel ausgeführt werden. Dies bedeutet, dass nur der Aufruf zum Ausführen der Prozedur gesendet wird, anstatt dass jede einzelne Codezeile einzeln gesendet wird.

Beispiel

Der Syntax einer Stored Procedure lautet wie gefolgt:

```
CREATE PROCEDURE procedure_name
AS
sql_statement
GO;
```

Dabei wird „procedure_name" mit einem Namen für die Prozedur ersetzt, z.B. AlleBenutzer. Und das „sql_statement" wird mit einer Reihe von SQL-Abfragen ersetzt, z.B. SELECT * FROM User. Dies würde also so aussehen:

```
CREATE PROCEDURE AlleBenutzer
AS
SELECT * FROM User
GO;
```

Diese gespeicherte Prozedur würde man mit „EXEC *AlleBenutzer"* aufrufen.

Prepared Statements

Ein Prepared Statement (vorbereitete Anweisung) ist eine sogenannte vorbereitete Anweisung für ein Datenbanksystem. Im Gegensatz zu gewöhnlichen Statements enthält es noch keine Parameterwerte. Stattdessen werden der Datenbank Platzhalter übergeben.

Vorteile

- Vorbereitete Anweisungen verhindern SQL-Injektion, da die dafür nötige Vermischung von SQL- und Hostsprache-Code nicht mehr ohne weiteres möglich ist.
- Darüber hinaus werden die Parameter beim Eintragen zunächst vom Datenbanksystem auf Gültigkeit geprüft bevor sie verarbeitet werden
- Das Verwenden von Prepared Statement bringt den Vorteil mit, dass durch die Vorverarbeitung der Abfragen eine schneller Verarbeitung der Abfrage möglich ist.

Beispiel

In PHP würde so eine vorbereitete Anweisung in etwa so aussehen:

```php
<?php
$stmt = $dbh->prepare("SELECT user, password FROM tbl_user WHERE
(user=:user)");
$stmt->bindParam(':user', $user);

// eine Zeile abfragen
$user = 'Alice';
$stmt->execute();

// eine weitere Zeile mit anderen Werten abfragen
$user = 'Bob';
$stmt->execute();
?>
```

Dabei wird „:user” als Platzhalter verwendet. In den folgenden Zeilen wird dann der Username abgefragt und dabei ergibt sich Alice und Bob. Diese werden mit $stmt->execute(); als für den Platzhalter eingesetzt.

Quellen

https://www.w3schools.com/sql/sql_stored_procedures.asp
https://glossar.hs-augsburg.de/Prepared_Statement