1. Php is supposed to return all reousrces by garbage collector (GC) when the page ends. In other words, mysql open automatically closes at the end of the page. then you can ask why you need to close it. When you want to break the connection and use another connection.

in time-consuming page treatment, db connection for resource duplication waste when used only in the first half. you can see it like this. in other words, it doesn't matter if the page you're processing is very fast, or if you don't have a lot of resource usage, you don't have to close it, but that's a general view, and it might be more efficient to close it according to logic. however, this efficiency is only a resource view, and if you add other costs, it may be more inefficient to pay attention to close.

1. IT IS $\_POST TO USE WHEN YOU WANT TO TAKE FORM DATA FROM THE CLIENT AND PUT THE VALUE IN THE VARIABLE WITHIN THE PHP SCRIPT OF THE SERVER, AND THIS IS ALSO A VARIABLE. THE PURPOSE IS TO BE A GLOBAL VARIABLE THAT CONTAINS FORM DATA. If the dollar ($) mark is preceded, it is called the PHP variable. There is a dollar mark in front of the $\_POST, so of course it is a variable and an array. If the user enters multiple form values and one of the form data isuserid (for example, name="userid"), the server's phpscript $user\_id = $\_POST ['userid']; I will write it in a way. Let's not forget that global variables and arrays are.

Example)

**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="utf-8">**

**<title>input</title>**

**</head>**

**<body>**

**<form method="post" action="check.php">**

**<p>**

**Name**

**<input type="text" name="name">**

**</p>**

**<input type="submit" value="submit">**

**</form>**

**</body>**

**</html>**



**<!DOCTYPE html>**

**<html>**

**<head>**

**<meta charset="utf-8">**

**<title>check</title>**

**</head>**

**<body>**

**<?php**

**$name = $\_POST['name'];**

**?>**

**<p>Name <?php echo $name ?></p>**

**</body>**

**</html>**



1. Sorry. I don’t understand this question.
2. The database is also used to store and manage data in these tables to configure one or more tables. Use MySQL's CREATE TABLE statement to create a new database. For example, let's write php that creates a reservation table using 4 fields.

**<?php**

**$servername = "localhost";**

**$user = "subin";**

**$password = "";**

**$dbname = "testDB";**

**$connect = mysqli\_connect($servername, $user, $password, $dbname);**

**if (!$connect) {**

**die("connection failed : ".mysqli\_connect\_error());**

**}**

**① $sql = "CREATE TABLE Reservation**

**(**

**ID INT PRIMARY KEY,**

**Name VARCHAR(30) NOT NULL,**

**ReservDate DATE NOT NULL,**

**RoomNum INT**

**)";**

**if (mysqli\_query($connect, $sql)) {**

**echo "table created!";**

**} else {**

**echo "failed to create table! : ".mysqli\_error($connection);**

**}**

**mysqli\_close($connection);**

**?>**

1. It seems to be related to SQL injection. SQL Injection is an act where amalicious user exposure a security vulnerability to inject and extreme SQL states to manipulate the database to be avoided.
2. The reason for this is simple. This is to prevent users other than administrators from arbitrarily tampering with or deleting data.
3. One of method is create a filtering method to filter out malicious code unconditionally.

special characters to filter: ' (hole quotation marks) ;(colon), --(comment), blank, etc.

I’m going to write an example of making a function using a regular expression.

Example)

function SQLFiltering($str){

// Code to prepare for hacking attacks.

$str=preg\_replace("/\s{1,}1\=(.\*)+/","",$str); // Remove 1 = 1 if there's a gap.

$str=preg\_replace("/\s{1,}(or|and|null|where|limit)/i"," ",$str); // If there's an or, and after a break, remove it.

$str = preg\_replace("/[\s\t\'\;\=]+/","", $str); // Remove spaces, tabs, and special characters.

return $str;

}

This may not be the perfect way, but from what I have studied, it can be a simple way to prevent SQL injection.

8. Cross-Site-Scripting, an attack that inserts malicious HTML tags or scripts into dynamically created web pages. (Another injection)

There are three types of XSS. Reflected Xss, Stored Xss, and Dom Xss.

In order to understand the explanation below more easily, it is convenient to think that there is an attacker and a victim, and there is a server in between. It is often used for sniffing cookies or displaying malicious sites or notification windows.

Reflected XSS

Translated directly, it is illuminating XSS, reflecting XSS.

It is a method of attacking the client by convincing the user to click on the url with the malicious script. It is said that a lot of this is done through direct messages or web bulletin boards. This method is often blocked by the browser itself, so it is difficult to attack.

Stored XSS

Some sites call it Persistent. Although similar to Reflected XSS, Reflected XSS is an attack method that induces users to click on a URL containing a malicious script.

DOM XSS

When the victim browser parses the HTML page, the attack script is executed as part of the DOM... So What is DOM?

DOM is, Document Object Model, the document object model. I went around several websites and summarized the definition is How web browsers read HTML pages I think that.. DOM XSS will be added and refined later to secure it as it is studied further.

The biggest reason is that XSS filtering is turned off from the client's point of view, or malicious phrases are not filtered from the server's point of view. It is caused by poor filtering and is very similar to SQL injection in that it injects malicious statements. However, the difference is that SQL injection attacks the server's DB, while XSS is an attack that occurs on other clients, that is, users who use the site. After researching like this, I started to understand little by little that the attacker mentioned above would be at a disadvantage if the victim was Chrome. Chrome and IE are said to have a security function that internally protects XSS. (There is Firefox, but they say it doesn't work well).

So How to compensate for XSS vulnerabilities

First of all, on the server side, it is important not to store any sensitive information in cookies. This is because cookie sniffing may result in the disclosure of sensitive information. It is also recommended to filter and set the characters that can be entered by the user. Blacklist bypass method popularized? Because it is an existing attack method, it is also good to block the whitelist method. Do not write text in html format where the user is inputting. On the user side... let's turn on the XSS filter unless it's for developers or security checks. Do not turn it off because the default value is on. Also, it is recommended to update frequently when a web browser update is available.