What is PHP?

PHP (recursive acronym for PHP: Hypertext Preprocessor)

PHP is a [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting) language designed primarily for [web development](https://en.wikipedia.org/wiki/Web_development) but also used as a [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language).  Originally created by [Rasmus Lerdorf](https://en.wikipedia.org/wiki/Rasmus_Lerdorf" \o "Rasmus Lerdorf) in 1994, the PHP [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) is now produced by The PHP Development Team.

SUPERGLOBALS

Superglobals — Superglobals are built-in variables that are always available in all scopes

Several predefined variables in PHP are "superglobals", which means they are available in all scopes throughout a script. There is no need to do **global $variable;** to access them within functions or methods.

These superglobal variables are:

* $\_SERVER – the $\_SERVER superglobal represents data available to a PHP script from the Web server itself.

|  |  |
| --- | --- |
| KEY | VALUE |
| $\_SERVER['argv'] | This variable returns the query string that is associated with a request. |
| $\_SERVER['DOCUMENT\_ROOT'] | The path to the application root. |
| $\_SERVER['HTTP\_ACCEPT'] | The value of the Accept header. |
| $\_SERVER['HTTP\_ACCEPT\_CHARSET'] | The value of the Accept-Charset header. |
| $\_SERVER['HTTP\_ACCEPT\_ENCODING'] | The value of the Accept-Encoding header. |
| $\_SERVER['HTTP\_ACCEPT\_LANGUAGE'] | The value of the Accept-Language header. |
| $\_SERVER[HTTP\_CONNECTION'] | The value of the Connection header. |
| $\_SERVER['HTTP\_HOST'] | The value of the Host header. |
| $\_SERVER['HTTP\_REFERER'] | The value of the Referer header. |
| $\_SERVER['HTTP\_USER\_AGENT'] | The value of the User-Agent header. |
| $\_SERVER['HTTPS'] | The value 'https' if the request was made by using the https transport. |
| $\_SERVER['REMOTE\_ADDR'] | The IP address of the client that is making the request. |
| $\_SERVER['REMOTE\_HOST'] | The host name of the client that is making the request. |
| $\_SERVER['REMOTE\_PORT] | The port number of the client that is making the request. |
| $\_SERVER['SCRIPT\_FILENAME'] | The file name of the script that is being invoked. |
| $\_SERVER['SCRIPT\_NAME'] | The name of the script that is being invoked. |
| $\_SERVER['SERVER\_PORT'] | The port number that the server accepted the request on. |
| $\_SERVER['REQUEST\_METHOD'] | The HTTP method of the request. |
| $\_SERVER['REQUEST\_URI'] | The URI associated with the HTTP request. |
| $\_SERVER['REQUEST\_TIME'] | The time stamp when the request was dispatched to the script. |

* $\_GET – the $\_GET superglobal represents data sent to the PHP script in a URL.
* $\_POST – the $\_POST superglobal represents data sent to the PHP script via HTTP POST. This is normally a form with a method of POST.
* $\_FILES – the $\_FILES superglobal represents data available to a PHP script from HTTP POST file uploads. Using $\_FILES is the currently preferred way to handle uploaded files in PHP.
* $\_COOKIE – the $\_COOKIE superglobal represents data available to a PHP script via HTTP cookies.
* $\_SESSION – The $\_SESSION superglobal represents data available to a PHP that has previously been stored in a session.
* $\_REQUEST – the $\_REQUEST superglobal is a combination of $\_GET, $\_POST, $\_COOKIE
* $\_ENV – the $\_ENV superglobal represents data available to a PHP script from the environment in which PHP is running.

SESSION FUNCTIONS

* Session\_abort – discard session array changes and finish session
* Session\_cache\_expire – return current cache expire
* Session\_cache\_limiter – get and/or set the current cache limiter
* Session\_commit – alias of session\_write\_close
* Session\_create\_id – create new session id
* Session\_decode – decodes session data from a session encoded string
* Session\_destroy – destroys all data registered to a session
* Session\_encode – encodes the current session data as a session encoded string
* Session\_gc – perform session data garbage collection
* Session\_get\_cookie\_params – get the session cookie parameters
* Session\_id – get and/or set the current session id
* Session\_is\_registered – find out whether a global variable is registered in a session
* Session-module\_name – get and/or set the current session module
* Session\_name – get and/or set the current session name
* Session\_regenerate\_id – update the current session id with a newly generated one
* Session\_register\_shutdown – sessions shutdown function
* Session\_register – register one or more global variables with the current session
* Session\_reset – re-initialize session array with original values
* Session\_save\_path – get and/or set the current session save path
* Session\_set\_save\_handler – set user-level session storage functions
* Session\_start – start new or resume existing session
* Session\_status – returns the current session status
* Session\_unregister – unregister a global variable from the current session
* Session\_unset – free all session variable
* Session\_write\_close – write session data and end session

Java Server Pages

* Simply an HTML web page that contains additional bits of code that execute application logic to generate dynamic content. 
* Java Server Pages Actions (JSP tags) perform a variety of functions and extend the capabilities of JSP.
* Java Server Pages Actions use XML-like syntax, and are used to manage JavaBeans component.
* Directives are instructions that are processed by the JSP engine when the page is compiled to a servlet.
* Directives are used to set page-level instructions, insert data from external files, and specify custom tag libraries.
  + <%@ %>
* Motivation
  + It is typically a good idea to separate business logic from presentation concern
    - Allows modern web development teams to be divided up into programmers and web page authors / designers.
    - Fosters component reuse (e.g. the same data object can be consumed by user agents of varying capabilities and needs)
  + Servlets can be very powerful for programming business logic, but are very awkward to use when generating static (i.e. template) content.
  + (X)HTML marked-up documents are very convenient for static content generation but cannot be used to program business logic (or generate dynamic content arising from data produced by the business logic).
* Features
  + Text-based document capable of generating both static and dynamic content (typically intermixed).
  + Mark-up based document syntax (JSP-style or XML-style), combining (X) HTML elements as well as standard and custom JSP elements; thus, web page authors can feel right “at home” with the mark-up syntax.
  + Embedded Java Coding support via “scriptlets”.
    - <% %>
  + Template text are converted into JSPWriter
* Components
  + Template (i.e. static) text
  + JSP elements
    - Directives
      * <%@ page ContentType=”text/html” pageEncoding=”UTF-8” %>
      * <%@ page import=”java.util.Random” %>
        + autoFlush
        + buffer
        + contentType
        + errorPage
        + extends
        + import
        + info
        + isELIgnore
        + isErrorPage
        + isThreadSafe
        + language
        + pageEncoding
        + session
      * <%@ taglib uri=”http://java.sun.com/jsp/jstl/core” prefix=”c” %>
        + prefix
        + taglib
        + uri
    - Scripting Elements
      * Declarations
        + <%! int a = 100; %>
        + <%! int square(int n) { return n\*n ; } %>
      * Expressions
        + <% String s = new java.util.Date().toString(); %>
      * Scriptlets
        + <% for(int i = 0; i < 10 ; i++) { out.println(i); } %>
    - Actions
      * Standard actions
        + <jsp:directive.include>, <jsp:directive.page>
        + <jsp:declarations>
        + <jsp:scriptlet>
        + <jsp:expression>
        + <jsp:include>, <jsp:forward>
        + <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty>
        + <jsp:plugin>, <jsp:param>, <jsp:params>, <jsp:fallback>
        + <jsp:element>, <jsp:attribute>, <jsp:body>
        + <jsp:text>
      * Custom Actions (JSTL)
        + JSTL, user-written custom tag libraries
      * Expression Language (EL)
        + ${ }
      * Implicit Scripting Objects
        + request, response, out, pageContext
        + session, pageContext, application
        + config, page, exception
      * Implicit EL Objects
        + pageContext
        + pageScope
        + requestScope
        + sessionScope
        + applicationScope
        + param, paramValues
        + header, headervalues
        + cookie
        + initparam
      * Comments
        + <%-- this is a JSP comment --%>

JSP Standard Tag Library (JSTL)

* Set of custom JSP elements that provide various programmatic functionality via markup syntax
  + Core Tag Library
    - variable support, flow control, URL management
  + SQL Tag Library
    - Database connections, queries, updates
  + Internationalization Tag Library
    - Locate setting, message bundling, number formatting, date formatting
  + XML
    - Core XML processing, flow control, transformation
  + JSTL Function
    - String functions, collection lengths
* In addition to the JSTL, developers can also create their own tag libraries for commonly occurring tasks

OWASP

* The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted.

OWASP Top 10 Application Security Risks – 2017

* A1 – Injection
  + Injection flaws, such as SQL, OS, XXE, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
* A2 – Broken Authentication and Session Management
  + Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities (temporarily or permanently).
* A3 – Cross-Site Scripting (XSS)
  + XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user supplied data using a browser API that can create JavaScript. XSS allows attackers to execute scripts in the victim’s browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
* A4 – Broken Access Control
  + Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users’ data, change access rights, etc.
* A5 – Security Misconfiguration
  + Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
* A6 – Sensitive Data Exposure
  + Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
* A7 – Insufficient Attack Protection
  + The majority of applications and APIs lack the basic ability to detect, prevent, and respond to both manual and automated attacks. Attack protection goes far beyond basic input validation and involves automatically detecting, logging, responding, and even blocking exploit attempts. Application owners also need to be able to deploy patches quickly to protect against attacks.
* A8 – Cross-Site Request Forgery (CSRF)
  + A CSRF attack forces a logged-on victim’s browser to send a forged HTTP request, including the victim’s session cookie and any other automatically included authentication information, to a vulnerable web application. Such an attack allows the attacker to force a victim’s browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
* A9 – Using Components with Known Vulnerabilities
  + Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
* A10 – Underprotected APIs
  + Modern applications often involve rich client applications and APIs, such as JavaScript in the browser and mobile apps, that connect to an API of some kind (SOAP/XML, REST/JSON, RPC, GWT, etc.). These APIs are often unprotected and contain numerous vulnerabilities.