♥-lichen Glückwunsch!

Wenn Du dieses Quiz liest, hast Du die erste Runde bereits überstanden. Nun gilt es, Dein Fachwissen unter Beweis zu stellen:

Bist Du der „Backend-Typ“ oder der „Frontend-Typ“? Der Pragmatiker oder der Theoretiker? Alles zugleich? Oder ganz anders? Das Ziel sind *nicht* 100% richtige Antworten – sondern dass wir verstehen, wie Du an technische Fragestellungen herangehst. Denn je besser uns dies gelingt, desto präziser können wir Dir eine Stelle auf den Leib schneidern, die optimal zu Dir passt. Und Hand aufs Herz: bitte nicht schummeln, denn im Vorstellungsgespräch sprechen wir ja evtl. auch über diese Themen… ;-)

Je nach Frage kannst Du ankreuzen oder frei antworten und zusätzlich auch Anmerkungen zur Frage oder Antwort notieren. Tipps: Beachte bitte, dass manchmal auch mehrere Antworten zutreffen. Wenn Du einen Favoriten hast, erwähne das ruhig!

Viel Erfolg!

PHP-Strings …

* 1. … sind PHP-Objekte
  2. … sind intern Unicode-kodiert, daher können die Zeichen eines Strings aus fast jedem Alphabet der Welt stammen
  3. … werden als Wert („per value“) statt als Referenz an Methoden übergeben, können also den Stack überlaufen lassen
  4. … belegen Speicherplatz, der später automatisch wieder freigegeben wird (ähnlich einem „Garbage Collector“)
  5. … können maximal 1 MB (1.048.576 Byte) Daten beinhalten (abhängig vom „memory\_limit“ in der php.ini)

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

In PHP kann eine Klasse direkt … | Bonusaufgabe: welches andere, ähnliche Konzept kennst Du noch?

* 1. … von mehreren Klassen ableiten, aber nur ein Interface implementieren
  2. … von mehreren Klassen ableiten und mehrere Interfaces implementieren
  3. … von einer Klasse ableiten oder aber ein Interface implementieren
  4. … von einer Klasse ableiten und gleichzeitig beliebig viele Interfaces implementieren

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Abstrakte Klassen …

* 1. … beinhalten ausschließlich Methoden-Rümpfe („Signaturen“), aber keine Methoden-Implementierungen
  2. … können nicht instanziiert werden
  3. … müssen beim Ableiten um fehlende Methoden-Implementierungen ergänzt werden (man hat keine andere Wahl)
  4. … dürfen keine statischen Methoden beinhalten
  5. … können nicht abgeleitet werden

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Um „SQL-Injections“ zu vermeiden, sollte man…

* 1. … alle Nutzereingaben in das Standardinterface „INonInjectable“ casten
  2. … sicherstellen, dass die php.ini-Einstellung „prevent\_injections“ auf „1“ oder „True“ steht
  3. … wenn möglich „prepared statements“ einsetzen
  4. … alle Nutzereingaben filtern und escapen

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Was macht der folgende Ausdruck: /^\d\d\.\d\d\.[0-9]{4}/i ?

* 1. Er verursachte einen Fehler, denn es handelt sich dabei um keine gültige Funktion
  2. Er prüft primitiv, ob etwas mit einem deutsch formatierten Datum beginnt
  3. Er ermittelt alle zu findenden, deutsch formatierten Zahlen in einem Text
  4. Er sucht nach 4-stelligen Postleitzahlen

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Was gibt der folgende Code aus? Bonusaufgabe: wie kann man den Code besser schreiben?

class hello {

private $\_sGreeting = "Hello World";

function printGreeting() {

echo $this->\_sGreeting;

}

}

hello::printGreeting();

* 1. Der PHP-Interpreter wirft einen Fehler, weil die Funktion nicht statisch ist
  2. „Hello World“
  3. Es wird nichts ausgegeben, es wird aber auch kein Fehler geworfen
  4. Der PHP-Interpreter wirft einen Fehler, weil auf ein privates Attribut zugegriffen wird

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Du möchtest Benutzerpasswörter sicher in einer Datenbank speichern, um eine Benutzer-Anmeldemaske zu programmieren – welche(n) der folgenden Algorithmen würdest Du dafür nutzen? Bonusaufgabe: warum?

1. BASE64
   1. RSA-768
   2. AES-256
   3. SHA-256
   4. MD5

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Joine die SQL-Tabellen VORLESUNG und HOERSAAL über die Felder VORLESUNG.HOERSAALID und HOERSAAL.ID. Vermeide vorlesungslose Hörsäle und hörsallose Vorlesungen. Sortiere nach VORLESUNG.TITEL und selektiere alle Felder. Bonusaufgabe: welche(n) Index / Indizes empfiehlst Du?

Klicke oder tippe hier, um Text einzugeben.

Erweitere die vorherige Abfrage um die Ermittlung der Anzahl der teilnehmenden Studenten pro Vorlesung aus der weiteren Tabelle STUDENT sowie der Kreuztabelle VORLESUNG2STUDENT (und führe das obige Namensschema der Felder fort):

Klicke oder tippe hier, um Text einzugeben.

Auf welche Weise(n) kannst Du die folgende Methode dazu zwingen, lediglich Instanzen der Klasse Greeting anzunehmen?

function printGreeting($oGreet) {

echo $oGreet->getGreeting();

}

Klicke oder tippe hier, um Text einzugeben.

Wie kannst Du „Session Fixation“ vermeiden?

* 1. ini\_set('session\_prevent\_fixation', '1')
  2. session\_regenerate\_id()
  3. explode()
  4. session\_cache\_expire()
  5. Indem man alle Nutzerdaten in der Session escapt
  6. Unnötig seit PHP 5.3, weil seitdem in der php.ini standardmäßig „session\_prevent\_fixation = 1“ gesetzt ist

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

PHP-Arrays …

* 1. … sind keine Objekte
  2. … implementieren das IEnumerable-Interface
  3. … implementieren das ArrayObject-Interface
  4. … werden (ggf. speichplatz- und zeitaufwendig) kopiert, wenn sie an Funktionen / Methoden übergeben werden
  5. … können zu einem Stack-Overflow führen, wenn sie zu oft an Funktionen / Methoden übergeben werden

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Was lässt sich über diese Funktion sagen?

function factorial($n, $r=1) {

return !$n ? $r : factorial($n - 1, $r \* $n);

}

* 1. PHP 7.0 oder neuer interpretiert dies als Schleife, daher kann kein Stack-Overflow auftreten („Tail-Call Optimization“)
  2. Ungültige Eingabewerte sind nahezu unmöglich bzw. werden effektiv verhindert
  3. Der PHP-Interpreter wirft einen Fehler, da sich eine PHP-Funktion niemals selbst aufrufen darf
  4. Aufruf mit großen Werten für $r führt zu einem Buffer-Overflow
  5. Aufruf mit großen Werten für $n führt zu einem Stack-Overflow
  6. Aufruf mit negativen Werten für $n führt zu einem Stack-Overflow
  7. Aufruf mit großen Werten für $n führt zu INF als Rückgabe
  8. !$n ist ok, weil PHP nicht zwischen Integer-0 und Boolean-False differenziert

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Du schreibst eine Anwendung, die sensible Daten mit dem BND austauscht. Wie stellst Du beim Datentransfer sicher, dass diese Daten nicht ausgespäht oder unerkannt manipuliert werden?

* 1. Hash der Daten erstellen und mit Deinem private Key verschlüsseln, Daten mit dem Public Key des BND verschlüsseln
  2. Hash der Daten erstellen und mit dem private Key des BND verschlüsseln, Daten mit Deinem Public Key verschlüsseln
  3. Daten mit Deinem private Key verschlüsseln
  4. Daten mit dem Public Key des BND verschlüsseln

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Der durch Objekte und andere Variablen belegte Hauptspeicher …

* 1. … muss explizit wieder freigeben werden, damit dem Programm nicht irgendwann der Speicher ausgeht („Heap“)
  2. … kann grundsätzlich nie zur Neige gehen, weil der Rechner notfalls Daten auf die Festplatte auslagert („swappt“)
  3. … steigt bei langläufigen Prozessen immer stärker an und kann ein Programm zum Absturz bringen
  4. … wird irgendwann vollautomatisch durch einen versteckten Mechanismus deallokiert
  5. … kann durch Nutzung von Arrays statt vieler Variablen minimiert werden

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

PHP-Sessions …

* 1. … können maximal 1 MB (1.048.576 Byte) Daten aufnehmen
  2. … sind unverwundbar gegenüber „Session-Highjacking“, weil Sessiondaten auf dem Server (und nicht im Browser) sind
  3. … funktionieren ohne weiteres i. d. R. nicht auf Webserver-Clustern (Bonusaufgabe: warum nicht? Bzw. warum doch?)
  4. … speichern ihre Daten ($\_SESSION[…]) an genau einer Stelle, im Hauptspeicher (RAM) als Teil des PHP-Prozesses
  5. … sind predestiniert für alle View-Parameter wie z. B. Filter- und Sortierkriterien, Pagination etc.
  6. … können nur in Codezeilen gestartet werden, die vor der ersten Ausgabe stehen (per „echo“ o. ä.)
  7. … funktionieren nicht ohne Cookies, weil alle Sessiondaten requestübergreifend im Cookie persistiert werden

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Was lässt sich über den Codestil dieses Ausschnittes im Allgemeinen sagen? Dieser Codestil …

<div id="price"><?php $myValue=getPrice() \* 1.19; echo $myValue . " €"; ?></div>

* 1. … funktioniert, sofern in der php.ini „legacy\_inline\_html = 1“ gesetzt ist (standardmäßig ab PHP 7.0)
  2. … funktioniert, die Funktion getPrice() darf aber nicht als „private“ deklariert sein
  3. … ist effizient, weil die Abhängigkeit zwischen PHP-Backend-Entwickler und Webdesigner reduziert wird
  4. … ist problematisch, weil im Frontend nicht gerechnet werden sollte
  5. … ist problematisch, weil zwangsläufig das Risiko für so genannte „Cross-Site-Scripting“-Angriffe (XSS) erhöht wird
  6. … macht Code kompakter und somit leichter wart- und testbar
  7. … reduziert die Testbarkeit des Codes
  8. … funktioniert überhaupt nicht mehr (abgeschafft seit PHP 7.0)
  9. … ist problematisch, weil Funktionen zugunsten von Methoden ab PHP 8.0 abgeschafft werden sollen
  10. … funktioniert, aber reduziert möglicherweise die Rendering-Geschwindigkeit des Webbrowsers

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Welches Entwurfsmuster („Design Pattern“) wurde hier versucht zu implementieren und wie würdest Du es reparieren?

class QuizQuestion {

public $\_myInstance = null;

public function \_\_construct() {}

private function getMyInstance() {

if ($this->\_myInstance) {

$this->\_myInstance == new self($this);

}

return $this->\_myInstance;

}

}

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Welche Vorgehensweise(n) prüft / prüfen, ob „@“ in $email enthalten ist? Bonusaufgabe: wie würdest Du es lösen?

* 1. if (strpos($email, "@") > 0) {…}
  2. if (instr($email, "@") !== false) {…}
  3. if (stripos("x" . $email, "@") > 0) {…}
  4. if (stripos($email, "@") >= 0) {…}
  5. if (stripos($email, "@") >== 0) {…}

Anmerkungen zur Frage oder Antwort? Einfach hier aufschreiben…!

Wenn man ungültige Formularinhalte (z. B. ungültige E-Mail-Adressen) per jQuery vermeiden möchte, muss man beachten, …

* 1. … dass man vorher die jquery.js inkludiert, wenn man möchte, dass alles bei 100% der Seitenbesucher lauffähig ist
  2. … dass man zusätzlich serverseitigen JavaScript-Code schreiben sollte, für den Fall dass die clientseitige fehlschlägt
  3. … dass man außerdem auch eine Validierung z. B. in PHP programmieren sollte
  4. … dass man den Bedienkomfort durch so genannte „Input Types“ zusätzlich steigern kann
  5. … dass man zur Minimierung des Entwicklungsaufwandes so genannte „reguläre Ausdrücke“ nutzt
  6. … dass man das <form…validation="1"…> nicht vergisst

Du hast Anmerkungen zur Frage oder zu Deiner Antwort? Dann schreibe Sie einfach hier auf!

Welche der folgenden Aussagen über „Call-By-Value“ bzw. „Call-By-Reference“ ist / sind in PHP korrekt?

* 1. PHP nutzt für alle Parameter von Funktionen / Methoden standardmäßig Call-By-Value
  2. PHP nutzt für alle Parameter standardmäßig Call-By-Value, für Objekte aber Call-By-Reference
  3. Es macht keinen Unterschied, ob Objekte per Call-By-Value oder Call-By-Reference übergeben werden
  4. Werden Objekte per Call-By-Value übergeben und in der Funktion / Methode geändert, ändert sich nur eine Kopie
  5. PHP-Arrays werden immer mittels Call-By-Value übergeben

Du hast Anmerkungen zur Frage oder zu Deiner Antwort? Dann schreibe Sie einfach hier auf!

Du schreibst eine Backend-Funktion zur Mittelwert-Berechnung. Wie behandelst Du Fehler (z. B. nicht-numerische Eingaben)?

* 1. Ich gebe per „echo …“ detaillierte Fehlerinformationen, um die spätere Fehlersuche zu vereinfachen
  2. Ich werfe eine PHP-Exception, die der Aufrufer fangen und auswerten kann, sofern gewünscht
  3. Ich gebe 0 zurück, weil ich dem Aufrufer so das komplizierte Exception-Handling erspare
  4. Ich werfe (wie in PHP üblich) einen PHP-Fehler mittels trigger\_error()

Du hast Anmerkungen zur Frage oder zu Deiner Antwort? Dann schreibe Sie einfach hier auf!

Abschließend eine Quizfrage, die gar keine Quizfrage ist – aber nicht weniger einfach: welche Themen magst Du, welche nicht? Auch hier gilt: bitte Hand auf’s Herz, denn auch im Vorstellungsgespräch sprechen wir vielleicht über diese Themenbereiche. Gerne kannst Du auch das Freitextfeld benutzen, um Deine Antwort zusätzlich etwas zu begründen!

* 1. Frontend-Entwicklung (Masken / Dialoge / Formulare, Eingabevalidierung, Navigationsfluss, Benutzerinteraktion, …)
  2. Backend-Entwicklung (ER-Modelle, Schnittstellen, Datenvalidierung, Datenpersistenz, Fachlogik, …)
  3. Konzeption (Kundenberatung, Architektur, Anforderungsaufnahme und -dokumentation, …)
  4. Dokumentation (Quellcode-Dokumentation, Handbücher, ReadMe‘s, …)
  5. Qualitätssicherung (Unit-Testing, interaktive Tests, Planung / Erarbeitung / Ausführung von Testplänen, …)
  6. Projektleitung (Zeitplan-Erstellung und -Überwachung, Reporting, …)
  7. Kreation (Grafikdesign, Textgestaltung, Content-Erarbeitung, …)
  8. Akquise (Erstberatung von anfragenden Neukunden, Zeitschätzung, Kostenkalkulation, Planung, …)
  9. Kommunikation (E-Mail-Verkehr mit Kunden, Ticket-Beantwortung, Telefonische Kundenberatung, …)

Du hast Anmerkungen zur Frage oder zu Deiner Antwort? Dann schreibe Sie einfach hier auf!

Geschafft! Nun noch dieses Dokument speichern und ab an’s eWorks-Jobs-Team: jobs@eworks.de. Wir freuen uns!