Cosa ci servirà

Un IDE. (VSC)

Java istallato nella nostra macchina

Alcune estensioni e l’essenziale di PHP che vedremo più avanti

La documentazione che verrà fornita

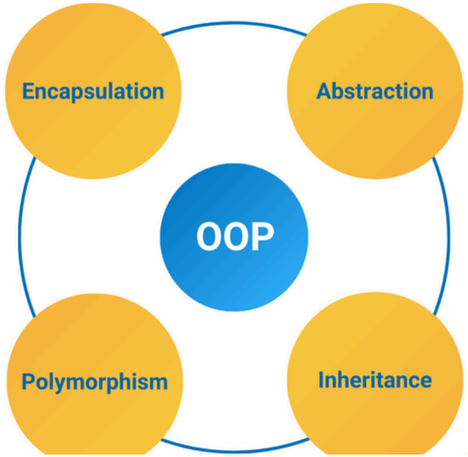
IDE Integrated Development Environment

Che cosa si intende per IDE?

Integrated Development Environment (IDE)

Un IDE, o ambiente di sviluppo integrato, è un software progettato per la realizzazione di applicazioni che aggrega strumenti di sviluppo comuni in un'unica interfaccia utente grafica.

OOP



In informatica, la programmazione orientata agli oggetti (in inglese object-oriented programming, in acronimo OOP), a volte chiamata semplicemente programmazione ad oggetti, è un paradigma di programmazione che permette di definire oggetti software in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi.

Particolarmente adatta nei contesti in cui si possono definire delle relazioni di interdipendenza tra i concetti da modellare (contenimento, uso, specializzazione), un ambito che più di altri riesce a sfruttare i vantaggi della programmazione ad oggetti è quello delle interfacce grafiche.

Vantaggi dell OOP

Alcuni dei vantaggi della programmazione OOP:

- **essa fornisce un supporto naturale alla modellazione software degli oggetti del mondo reale o del modello astratto da riprodurre;**
- **permette una più facile gestione e manutenzione di progetti di grandi dimensioni;**
- **l'organizzazione del codice sotto forma di classi favorisce la modularità e il riuso di codice.**

Carrateristiche dell OOP

ORA VEDEREMO UNO DEGLI ELEMENTI PRINCIPALI DELL'OOP:

LA CLASSE

Le classi definiscono dei tipi di dato e permettono la creazione degli oggetti secondo le caratteristiche definite nella classe stessa.

Grazie alle relazioni di ereditarietà, è possibile creare nuove classi a partire da quelle esistenti, estendendole con caratteristiche aggiuntive.

La classe è composta da:

- **attributi, ossia variabili e/o costanti che definiscono le caratteristiche o proprietà degli oggetti instanziabili invocando la classe; i valori inizializzati degli attributi sono ottenuti attraverso il cosiddetto costruttore;**

- **metodi, ossia procedure che operano sugli attributi.**

Caratteristiche di Unione (OOP in PHP)

- **Classi e Oggetti: PHP supporta le classi e gli oggetti, permettendo agli sviluppatori di definire strutture di classe con proprietà e metodi, e di istanziare oggetti da queste classi.**
- **Ereditarietà: PHP permette alle classi di ereditare proprietà e metodi da altre classi, facilitando il riutilizzo del codice e la creazione di gerarchie di classi.**
- **Incapsulamento: Le proprietà e i metodi possono essere definiti come pubblici, privati o protetti, permettendo un controllo dettagliato sull'accesso e proteggendo l'integrità dell'oggetto.**
- **Polimorfismo: PHP supporta il polimorfismo, principalmente attraverso l'overriding dei metodi (metodi nelle classi derivate che hanno lo stesso nome dei metodi nelle classi base).**
- **Interfacce e Classi Astratte: PHP supporta sia interfacce che classi astratte, consentendo agli sviluppatori di definire metodi che devono essere implementati nelle classi concrete.**

Caratteristiche di Differenza (PHP vs. OOP Puro)

- **Tipizzazione Dinamica: PHP è un linguaggio a tipizzazione dinamica, il che significa che il tipo di una variabile è determinato in fase di esecuzione. Questo può portare a meno rigidità e a maggior flessibilità rispetto ai linguaggi a tipizzazione statica, ma può anche introdurre errori difficili da tracciare.**
- **Maneggiamento degli Errori: La gestione degli errori in PHP tradizionalmente si basava su funzioni di gestione degli errori piuttosto che su eccezioni. Tuttavia, le versioni più recenti supportano pienamente le eccezioni, in linea con i principi OOP.**
- **Supporto OOP Non Completo: Nonostante le significative miglioramenti, alcune caratteristiche OOP come i metodi e le proprietà statiche finali o l'ereditarietà multipla non sono supportate direttamente in PHP (l'ereditarietà multipla può essere simulata usando le interfacce).**
- **Standardizzazione: A differenza di linguaggi OOP più rigorosi come Java o C#, PHP è meno standardizzato in termini di pratiche OOP, risultando in stili di programmazione più variabili tra gli sviluppatori.**

PHP quindi cos’è?

PHP, che è l'acronimo di "PHP: Hypertext Preprocessor", è un linguaggio di scripting lato server ampiamente utilizzato nello sviluppo web per generare pagine HTML dinamiche.

Originariamente creato da Rasmus Lerdorf nel 1994, PHP ha subito numerose evoluzioni e resta uno dei linguaggi di programmazione più popolari per il web.

PHP come configurarlo

1. **Installare Visual Studio Code:**
   - **Esatto (Come da sopra)**
2. **Installare l'estensione PHP:**
   - **Avvia Visual Studio Code.**
   - **Vai alla barra laterale sinistra e fai clic sull'icona delle "Estensioni"**
   - **Cerca "PHP" nell'elenco delle estensioni e installa l'estensione ufficiale di PHP.**
3. **Installare PHP:**
   - **Assicurati di avere PHP installato sul tuo computer. Puoi scaricarlo e installarlo dal sito ufficiale di PHP**
4. **Configurare il percorso di PHP in Visual Studio Code:**
   - **Dopo aver installato PHP, apri Visual Studio Code.**
   - **Vai su File > Preferenze > Impostazioni (o premi Ctrl+,).**
   - **Nella barra di ricerca delle impostazioni, cerca "php.executable".**
   - **Modifica il percorso per puntare all'eseguibile PHP. Ad esempio, su Window,s potrebbe essere qualcosa come "php.executablePath": "C:\\php\\php.exe".**

PHP come configurarlo con XAMPP

1. **Assicurati di avere XAMPP installato sul tuo computer:**
   - **Se non hai ancora installato XAMPP, puoi scaricarlo dal sito ufficiale e seguire le istruzioni di installazione appropriate per il tuo sistema operativo: Download XAMPP.**
2. **Avvia XAMPP e avvia i servizi Apache e MySQL:**
   - **Dopo aver installato XAMPP, avvia il programma e avvia i servizi Apache e MySQL facendo clic sui pulsanti "Start" accanto a ciascun servizio.**
3. **Configura Visual Studio Code:**
   - **Assicurati di avere Visual Studio Code installato sul tuo computer.**
   - **Installa l'estensione PHP per Visual Studio Code se non l'hai già fatto. Puoi farlo dalla sezione "Estensioni" di VSC cercando "PHP" e installando l'estensione ufficiale.**
   - **Configura il percorso dell'eseguibile PHP in Visual Studio Code. Per fare ciò, apri le impostazioni di Visual Studio Code (File > Preferenze > Impostazioni) e cerca "php.executablePath". Imposta il percorso dell'eseguibile PHP di XAMPP. Di solito, è qualcosa del genere: C:\xampp\php\php.exe.**

XAMPP

XAMPP è un pacchetto di software open source che fornisce un ambiente di sviluppo completo per la creazione e il test di applicazioni web localmente sul tuo computer. Il suo nome è un acronimo che sta per:

- **X: qualsiasi sistema operativo (Windows, Linux, Mac OS X)**
- **A: Apache HTTP Server**
- **M: MySQL database**
- **P: PHP**
- **P: Perl**

In breve, XAMPP ti fornisce tutto ciò di cui hai bisogno per iniziare a sviluppare e testare siti web dinamici senza dover configurare manualmente ciascun componente del server web.

Le funzioni principali di XAMPP:

I principali componenti di XAMPP:

1. **Apache HTTP Server: Apache è il server web più popolare al mondo, utilizzato per distribuire siti web e applicazioni web. In XAMPP, Apache è preconfigurato e pronto per essere utilizzato come server web locale.**
2. **MySQL: MySQL è un sistema di gestione di database relazionali (RDBMS) open source ampiamente utilizzato. Con XAMPP, MySQL è incluso e ti consente di creare e gestire database per le tue applicazioni web.**
3. **PHP: PHP è un linguaggio di scripting lato server ampiamente utilizzato per lo sviluppo di applicazioni web dinamiche. XAMPP include PHP, insieme a una serie di estensioni comuni, che ti consente di scrivere codice PHP e eseguirlo sul tuo server web locale.**
4. **Perl: Perl è un linguaggio di scripting versatile e potente utilizzato per una varietà di scopi, inclusi scripting di sistema, sviluppo web e analisi di testo. Anche se meno comune rispetto ad Apache, MySQL e PHP, XAMPP include anche Perl come opzione aggiuntiva.**

Le funzioni principali di PHP:

Andiamo a vedere una lista delle funzioni principali in PHP, insieme a una breve introduzione alle stesse:

- **Output su Schermo:**
  - **echo() - Stampa una o più stringhe.**
  - **print() - Stampa una stringa.**
- **Debugging e Informazioni sulle Variabili:**
  - **var_dump() - Visualizza informazioni sulla variabile, inclusi il tipo e il valore.**
- **Manipolazione delle Stringhe:**
  - **strlen() - Restituisce la lunghezza di una stringa.**
  - **substr() - Restituisce una sottostringa di una stringa.**
  - **strtoupper() - Converte una stringa in maiuscolo.**
  - **strtolower() - Converte una stringa in minuscolo.**
  - **str_replace() - Sostituisce tutte le occorrenze di una stringa con un'altra all'interno di una stringa.**
- **Manipolazione degli Array:**
  - **array() - Crea un array.**
  - **count() - Restituisce il numero di elementi in un array.**

Principali Concept di PHP:

- **Linguaggio di Scripting Lato Server:**
  - **PHP è un linguaggio di scripting interpretato lato server, progettato principalmente per lo sviluppo di applicazioni web dinamiche.**
- **Sintassi Simile a C:**
  - **La sintassi di PHP è influenzata da C, rendendola familiare a molti programmatori provenienti da linguaggi come C, C++, e Java.**
- **Ampia Disponibilità:**
  - **PHP è ampiamente supportato da molti provider di hosting web, rendendolo una scelta popolare per lo sviluppo di siti web dinamici e applicazioni web.**
- **Interoperabilità:**
  - **PHP può essere facilmente integrato con altri linguaggi di programmazione e tecnologie web come HTML, CSS, JavaScript, e database SQL.**
- **Orientamento agli Oggetti:**
  - **PHP supporta la programmazione orientata agli oggetti (OOP), consentendo agli sviluppatori di creare codice modulare, riutilizzabile e manutenibile.**

Principali Concept di PHP:

- **Ampia Libreria Standard:**
  - **PHP dispone di una vasta libreria standard che fornisce funzioni e classi predefinite per una varietà di scopi, come manipolazione delle stringhe, gestione dei file, accesso ai database, e altro ancora.**
- **Gestione delle Sessioni e dei Cookie:**
  - **PHP offre funzionalità integrate per gestire le sessioni utente e i cookie, consentendo agli sviluppatori di creare applicazioni web con funzionalità di autenticazione e personalizzazione.**
- **Gestione degli Errori:**
  - **PHP fornisce meccanismi per la gestione degli errori, inclusi avvisi, eccezioni e logging, per facilitare il debug e il monitoraggio delle applicazioni.**
- **Flessibilità:**
  - **PHP è un linguaggio flessibile che supporta diversi paradigmi di programmazione, tra cui procedurale, orientato agli oggetti e funzionale.**
- **Open Source:**
  - **PHP è un linguaggio open source distribuito con licenza PHP License, che consente agli sviluppatori di utilizzarlo liberamente, modificarlo e distribuirlo senza restrizioni.**

Integrazione con HTML

PHP è progettato per essere flessibile e facilmente integrabile all'interno del codice HTML.

Questo permette agli sviluppatori di aggiungere funzionalità dinamiche a pagine web statiche senza la necessità di modificare l'intera struttura del documento.

Il codice PHP viene scritto tra i tag <?php e ?>, che segnalano al server di interpretare il contenuto tra questi tag come codice PHP.

Questa capacità di integrarsi e interagire direttamente con l'HTML rende PHP estremamente potente per lo sviluppo di applicazioni web interattive.

Integrazione con HTML

PHP grazie alle sue carrateristiche specifiche può essere integrato direttamente all'interno di un documento HTML. I tag <?php e ?> delimitano il codice PHP dal resto dell'HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>Esempio PHP</title>
</head>
<body>
    <h1><?php echo "Ciao, mondo!"; ?></h1>
</body>
</html>
```

Integrazione con HTML

In PHP, è possibile eseguire una varietà di operazioni che interagiscono efficacemente con l'HTML per creare applicazioni web dinamiche.
Queste operazioni possono includere il controllo di flusso, la manipolazione di dati, la generazione di contenuti HTML dinamici, e la gestione delle interazioni utente.
Di seguito, esploreremo alcune delle operazioni più comuni che si possono realizzare con PHP e HTML.

1. Generazione di Contenuti HTML Dinamici PHP è estremamente utile per generare HTML dinamicamente. Questo permette di creare pagine web che si aggiornano in base ai dati forniti dall'utente o da altre fonti, come database.

2. Gestione del Controllo di Flusso PHP supporta diverse strutture di controllo (come if, else, while, for, foreach) che possono influenzare il flusso di esecuzione del programma e la generazione di contenuti specifici in base a condizioni diverse.

3. Interazione con Database Le operazioni sui database, come l'inserimento, l'aggiornamento, la cancellazione e la lettura di dati, sono comuni in PHP, soprattutto utilizzando MySQL come sistema di gestione del database.

4. Gestione delle Sessioni PHP può gestire sessioni per tracciare le informazioni degli utenti durante la navigazione di diverse pagine di un sito, consentendo di mantenere il loro stato tra una pagina e l'altra.

Le variabili in PHP

Le variabili in PHP sono identificate da un prefisso $ e non richiedono una dichiarazione di tipo esplicita grazie alla tipizzazione debole del linguaggio.

Questo significa che il tipo di una variabile è determinato automaticamente in base al contesto in cui viene utilizzata.

Questa caratteristica rende PHP molto flessibile, ma può anche portare a errori di programmazione meno prevedibili, motivo per cui gli sviluppatori devono essere attenti nella gestione dei tipi di dati.

Parametri e Valori di Ritorno

PHP supporta diversi tipi di dati che permettono agli sviluppatori di manipolare varie forme di informazioni. Comprendere questi tipi di dati è fondamentale per la programmazione efficace in PHP, poiché ogni tipo ha caratteristiche e usi specifici.
Tipi di Dati in PHP

1. Intero (Integer): Gli interi sono numeri senza decimali che possono essere positivi o negativi. PHP gestisce gli interi in base alla piattaforma (32-bit o 64-bit).

2. Punto Flottante (Float): Anche noti come "double", i float sono numeri che includono una componente decimale. Sono utilizzati quando è necessaria una maggiore precisione nei calcoli.

3. Stringa (String): Le stringhe sono sequenze di caratteri, usate per manipolare testi. In PHP, le stringhe possono essere definite usando sia apici singoli che doppi.

Parametri e Valori di Ritorno

4. Booleano (Boolean): Questo tipo di dato ha solo due valori possibili: true e false. È spesso utilizzato in condizioni e decisioni logiche.

5. Array: Gli array in PHP sono mappe ordinate che associano valori a chiavi. Gli array possono contenere elementi di diversi tipi di dati e sono estremamente versatili.

6. Oggetto (Object): Gli oggetti in PHP sono istanze di classi progettate per l'incapsulamento di dati e funzioni relative a quei dati. La programmazione orientata agli oggetti è una caratteristica chiave di PHP.

7. NULL: Un tipo speciale che ha un solo valore possibile: NULL. È utilizzato per rappresentare una variabile senza nessun valore valido.

8. Resource: Le risorse non sono un tipo di dato vero e proprio, ma sono riferimenti a risorse esterne, come file aperti o connessioni a database.

Parametri e Valori di Ritorno

```php
// Intero
$numero = 42;
echo $numero; // Output: 42

// Punto Flottante
$decimal = 10.5;
echo $decimal; // Output: 10.5

// Stringa
$testo = "Ciao mondo!";
echo $testo; // Output: Ciao mondo!

// Booleano
$vero = true;
$falso = false;
echo $vero;  // Output: 1 (true viene stampato come 1)
echo $falso; // Output non stampa nulla per false
```

```php
// Array
$array = array("mela", "banana", "ciliegia");
echo $array[1]; // Output: banana

// Oggetto
class Auto {
 public $colore = "rosso";
}
$miaAuto = new Auto();
echo $miaAuto->colore; // Output: rosso

// NULL
$niente = NULL;
var_dump($niente); // Output: NULL

// Resource
$file = fopen("test.txt", "r");
var_dump($file); // Output: resource(3) of type (stream)
```

Le Variabili in PHP

Le variabili in PHP sono prefissate con un simbolo $.
Non è necessario dichiarare il tipo di dato, poiché PHP è un linguaggio debolmente tipizzato:

```
1. <?php
2. $numero = 5;          // Variabile di tipo intero
3. $testo = "Hello";     // Variabile di tipo stringa
4. $float = 3.14;        // Variabile di tipo float
5. $array = [1, 2, 3];   // Variabile di tipo array
6. ?>
```

Sintassi basica in PHP

La sintassi di PHP è influenzata da linguaggi di programmazione come C, Java e Perl, il che la rende familiare a molti sviluppatori.

Utilizza strutture di controllo comuni come if-else, cicli for e while, e permette la definizione di funzioni.

Questa familiarità nella sintassi aiuta a ridurre la curva di apprendimento per i nuovi sviluppatori e facilita la scrittura di codice chiaro e mantenibile.

Sintassi Basica di PHP

La sintassi di PHP è influenzata da linguaggi come C, Java e Perl. Utilizza strutture di controllo simili:

```
<?php
// Condizionale if-else
if ($numero > 0) {
    echo "Positivo";
} else {
    echo "Non positivo";
}

// Ciclo for
for ($i = 0; $i < 3; $i++) {
    echo $array[$i] . " "; // Stampa gli elementi dell'array
}

// Funzioni
function saluta($nome) {
    return "Ciao " . $nome . "!";
}
echo saluta("Alice");
?>
```

Tipizzazione Debole in PHP

La tipizzazione debole di PHP consente la conversione automatica tra tipi di dati diversi, basata sul contesto.

Questo può semplificare lo sviluppo di script, eliminando la necessità di specificare tipi di dati e di eseguire conversioni esplicite.

Tuttavia, questa flessibilità può anche portare a comportamenti inaspettati e bug, specialmente in applicazioni complesse o in ambienti di sviluppo dove la precisione dei tipi è critica.

Tipizzazione Debole in PHP

PHP gestisce automaticamente le conversioni di tipo.
Ad esempio, se si sommano una stringa e un numero, PHP cercherà di convertire la stringa in numero:

```
1. <?php
2. $var1 = "4gatti";
3. $var2 = 3;
4. echo $var1 + $var2;    //Output sarà 7, perché "4gatti" diventa 4
5. ?>
```

Concatenazione di Stringhe

PHP gestisce la concatenazione di stringhe tramite l'operatore ‘.’ .

Questa caratteristica è particolarmente utile per la generazione dinamica di contenuti HTML o per la manipolazione di testi all'interno delle applicazioni.

A differenza di altri linguaggi che possono usare operatori come + per la concatenazione di stringhe, PHP utilizza un operatore specifico per chiarire che l'operazione riguarda stringhe, riducendo così la confusione e prevenendo errori comuni legati alla tipizzazione debole.

Concatenazione di Stringhe

In PHP, le stringhe possono essere concatenate utilizzando il punto . :

```
1. <?php
2. $stringa1 = "Ciao, ";
3. $stringa2 = "come stai?";
4. echo $stringa1 . $stringa2; // Output: "Ciao, come stai?"
5. ?>
```

Condizioni in php

Teoria delle Condizioni in PHP

1. if Statement: La dichiarazione if è la più semplice forma di controllo. Esegue un blocco di codice solo se la condizione specificata è vera.

2. else Clause: Spesso usata insieme a if. Se la condizione nell'if non è vera, il codice all'interno del blocco else verrà eseguito.

3. elseif (o else if): Permette di testare più condizioni in sequenza, ciascuna con il proprio blocco di codice da eseguire. È utile quando ci sono più scenari possibili e ognuno richiede un trattamento diverso.

4. Operatore ternario: È una versione compatta di un'istruzione if-else, che è molto utile per assegnazioni condizionali. La sintassi è condizione ? valore_se_vera : valore_se_falsa;.

5. switch Statement: Una switch statement consente di eseguire diversi blocchi di codice a seconda del valore di una variabile. È una buona alternativa a if-elseif-else quando si ha a che fare con molte condizioni basate sullo stesso valore.

Esempi di Condizioni in PHP

Esempio 1: Uso di if, else, e elseif

```php
<?php
$età = 20;
if ($età < 18) {
    echo "Non sei abbastanza grande.";
} elseif ($età == 18) {
    echo "Appena abbastanza grande!";
} else {
    echo "Sei sicuramente abbastanza grande.";
}
?>
```

Esempi di Condizioni in PHP

Esempio 2: Uso di if, else, e elseif

```
<?php
$età = 20;
$status = ($età >= 18) ? "adulto" : "minorenne";
echo "Sei un $status.";
?>
```

Esempi di Condizioni in PHP

Esempio 3: Uso di if, else, e elseif

```php
<?php
$giorno = "mercoledì";

switch ($giorno) {
    case "lunedì":
        echo "Inizia la settimana.";
        break;
    case "mercoledì":
        echo "Metà settimana.";
        break;
    case "venerdì":
        echo "Quasi weekend!";
        break;
    default:
        echo "Un altro giorno della settimana.";
}
?>
```

Esempi di Condizioni in PHP

Condizioni multiple con operatori logici

In PHP, le strutture condizionali possono essere ampliate per valutare più condizioni contemporaneamente tramite l'uso degli operatori logici come AND (&&) e OR (||). Questo è utile in scenari reali dove è necessario verificare contemporaneamente diverse condizioni per decidere il flusso di esecuzione del codice.

Nel primo esempio, il programma verifica se un individuo ha l'età necessaria, possiede la patente e risiede in Italia per determinare se può guidare legalmente nel paese.

Questo tipo di verifica è comune in applicazioni che necessitano di autorizzazione o validazione di certi requisiti.

Esempi di Condizioni in PHP

Condizioni multiple con operatori logici
In questo esempio, utilizziamo if per verificare più condizioni utilizzando gli operatori AND (&&) e OR (||):

```
<?php
$età = 25;
$haPatente = true;
$statoResidenza = "Italia";

if ($età >= 18 && $haPatente && $statoResidenza == "Italia") {
    echo "Puoi guidare in Italia.";
} elseif ($età >= 18 && !$haPatente) {
    echo "Non puoi guidare, ti serve la patente.";
} else {
    echo "Non soddisfi i requisiti per guidare.";
}
?>
```

Esempi di Condizioni in PHP

Nidificazione di if

La nidificazione di istruzioni if è una tecnica che permette di creare controlli condizionali più dettagliati e specifici all'interno di un altro controllo condizionale.

Questo approccio è particolarmente utile quando le decisioni dipendono da una serie di criteri che devono essere valutati in sequenza.
Nel secondo esempio, la capacità di un individuo di accedere a corsi avanzati dipende non solo dall'età ma anche dal punteggio ottenuto.

La nidificazione consente di gestire questa complessità in modo chiaro e strutturato.

Esempi di Condizioni in PHP

Nidificazione di if
Un esempio di if nidificati per verificare più livelli di condizioni:

```php
<?php
$punteggio = 85;
$età = 20;

if ($età > 18) {
    if ($punteggio > 90) {
        echo "Ottimo lavoro, sei ammesso al livello avanzato.";
    } elseif ($punteggio > 75) {
        echo "Buon lavoro, sei ammesso al livello intermedio.";
    } else {
        echo "Devi migliorare per passare al livello successivo.";
    }
} else {
    echo "I corsi avanzati sono per maggiorenni.";
}
?>
```

Esempi di Condizioni in PHP

Uso con array e funzioni

L'utilizzo di condizioni che integrano array e funzioni built-in di PHP consente di scrivere codice dinamico e flessibile.

Verificare le condizioni basate su elementi specifici di un array e utilizzare funzioni per la validazione (come filter_var per la validazione di email) aiuta a mantenere il codice sicuro e robusto.

Questo tipo di controllo è indispensabile in applicazioni web moderne dove la validazione dei dati di input è critica per la sicurezza e l'integrità dell'applicazione.

Esempi di Condizioni in PHP

Uso con array e funzioni
Verifica condizioni basate su elementi specifici di un array e l'utilizzo di funzioni:

```php
<?php
$utente = [
    'nome' => 'Marco',
    'età' => 30,
    'email' => 'marco@example.com',
    'isVerified' => true
];

if ($utente['età'] >= 18 && filter_var($utente['email'], FILTER_VALIDATE_EMAIL)) {
    if ($utente['isVerified']) {
        echo "Benvenuto, " . $utente['nome'] . ". Il tuo account è verificato.";
    } else {
        echo "Ciao, " . $utente['nome'] . ". Per favore verifica il tuo account.";
    }
} else {
    echo "Non sei idoneo per accedere a questo servizio.";
}
?>
```

Esempi di Condizioni in PHP

Operatore ternario

L'operatore ternario è una forma condensata di esprimere istruzioni condizionali if-else, che rende il codice più conciso e leggibile.

È particolarmente utile in situazioni dove si deve assegnare un valore a una variabile basato su una condizione semplice.
Questa forma ridotta può migliorare la leggibilità del codice evitando strutture verbose, specialmente quando le operazioni condizionali sono semplici e diretto.

Tuttavia, è importante usarlo con cautela per evitare di rendere il codice troppo criptico, specialmente in presenza di logiche condizionali complesse.

Esempi di Condizioni in PHP

Operatore ternario
L'operatore ternario è una forma compatta per esprimere un'istruzione if-else.
Può essere usato per rendere il codice più conciso:

```
1. <?php
2. $età = 20;
3. $messaggio = ($età >= 18) ? "Sei maggiorenne." : "Sei
   minorenne.";
4. echo $messaggio;
5. ?>
```

Esercizio di PHP con Condizioni

Problema: Devi creare uno script PHP che determina il prezzo del biglietto di un museo in base all'età del visitatore e se il giorno corrente è un weekend o meno.

- **I bambini sotto i 12 anni entrano gratis.**
- **Gli adolescenti (dai 12 ai 17 anni) pagano 5 euro durante i giorni feriali e 3 euro nei weekend.**
- **Gli adulti (dai 18 anni in su) pagano 10 euro durante i giorni feriali e 8 euro nei weekend.**

Il tuo script deve prendere due input:

- **L'età del visitatore.**
- **Un booleano che indica se oggi è un weekend (true se è weekend, false se non lo è).**
- **Il tuo script deve poi stampare il prezzo del biglietto basandosi su queste informazioni.**

Soluzione Esercizio di PHP con Condizioni

```php
<?php
// Input: età del visitatore e se oggi è weekend
$età = 15;  // Inserisci qui l'età del visitatore
$isWeekend = true;  // Imposta a true se è weekend, altrimenti false

// Determinare il prezzo del biglietto
if ($età < 12) {
    $prezzo = 0;
    echo "L'ingresso è gratuito.";
} elseif ($età >= 12 && $età <= 17) {
    if ($isWeekend) {
        $prezzo = 3;
    } else {
        $prezzo = 5;
    }
    echo "Il prezzo del biglietto è di €" . $prezzo;
} else {
    if ($isWeekend) {
        $prezzo = 8;
    } else {
        $prezzo = 10;
    }
    echo "Il prezzo del biglietto è di €" . $prezzo;
}
?>
```

- **Questo script utilizza una struttura di condizioni nidificate per valutare prima l'età del visitatore e poi, all'interno di ogni ramo età-specifico, controlla se è weekend per determinare il prezzo finale.**

Esercizio di PHP con Condizioni

Problema: Devi creare uno script PHP che calcola il costo energetico mensile per un'abitazione in base al consumo energetico, alla tipologia di abitazione e al giorno della settimana in cui si effettua la lettura del contatore.

Regole per il calcolo:

- **Il costo base per kWh è di 0.20 euro.**
- **Per le abitazioni di tipo "residenziale", si applica uno sconto del 10% sul costo totale.**
- **Per le abitazioni di tipo "commerciale", si applica un supplemento del 15% sul costo totale.**
- **Se la lettura del contatore avviene nel weekend, si applica un ulteriore sconto del 5% sul costo dopo gli altri aggiustamenti per incoraggiare le letture in giorni di minore attività.**

Lo script deve prendere tre input:

1. **kWh consumati nel mese.**
2. **Tipologia di abitazione ("residenziale" o "commerciale").**
3. **Un booleano che indica se la lettura è stata fatta durante il weekend (true se è weekend, false se non lo è).**

Soluzione Esercizio di PHP con Condizioni

```php
<?php
// Input: kWh consumati, tipo di abitazione, e se la lettura è avvenuta nel weekend
$kWh = 350;  // kWh consumati nel mese
$tipoAbitazione = "commerciale";  // Tipo di abitazione: "residenziale" o "commerciale"
$isWeekend = true;  // Lettura avvenuta nel weekend

$costoBasePerKWh = 0.20;  // Costo base per kWh
$costoTotale = $kWh * $costoBasePerKWh;  // Calcolo del costo totale iniziale

// Applicazione degli sconti o supplementi basati sulla tipologia di abitazione
if ($tipoAbitazione == "residenziale") {
    $costoTotale *= 0.90;  // Sconto del 10%
    echo "Tipo di abitazione: Residenziale. Applicato sconto del 10%. ";
} elseif ($tipoAbitazione == "commerciale") {
    $costoTotale *= 1.15;  // Supplemento del 15%
    echo "Tipo di abitazione: Commerciale. Applicato supplemento del 15%. ";
}

// Ulteriore sconto se la lettura è avvenuta nel weekend
if ($isWeekend) {
    $costoTotale *= 0.95;  // Ulteriore sconto del 5%
    echo "Lettura effettuata nel weekend. Applicato ulteriore sconto del 5%. ";
}

echo "Il costo energetico mensile è di €" . number_format($costoTotale, 2);
?>
```

Eccezioni Comuni in Condizioni PHP

- **Divisione per zero: Uno degli errori più comuni si verifica quando si tenta di dividere un numero per zero. PHP genera un warning, ma il risultato sarà INF o NAN, a seconda del contesto. È fondamentale controllare che il denominatore non sia zero prima di eseguire una divisione.**
- **Accesso a indici di array non esistenti: Accedere a un elemento di un array che non esiste può generare un notice di tipo "Undefined index". È buona norma verificare se un indice esiste con isset() o array_key_exists() prima di accedervi.**
- **Uso di variabili non inizializzate: Utilizzare una variabile non inizializzata in un'espressione condizionale può portare a comportamenti imprevisti e notice di "Undefined variable". Assicurarsi sempre di inizializzare le variabili.**

Pattern di Utilizzo e Best Practice

Pattern di Utilizzo e Best Practice

- **Uso di isset() e empty(): Quando si lavora con variabili che potrebbero non essere state inizializzate o potrebbero essere null, è consigliabile utilizzare isset() per verificare che una variabile sia stata definita e non sia null. Utilizzare empty() quando si vuole controllare anche che il valore non sia equivalente a false, zero, una stringa vuota, o null.**

- **Utilizzare operatori di confronto rigorosi: In PHP, è preferibile utilizzare gli operatori di confronto rigorosi (=== e !==) rispetto a quelli standard (== e !=). Questo perché gli operatori rigorosi confrontano sia il tipo che il valore, evitando conversioni implicite che possono portare a risultati inattesi.**

Pattern di Utilizzo e Best Practice

- **Gestione delle eccezioni: Se il tuo codice PHP è orientato agli oggetti, utilizzare blocchi try-catch per gestire le eccezioni in modo controllato. Ciò è particolarmente utile per operazioni che potrebbero fallire, come l'accesso a file, database, o risorse di rete.**

- **Logica condizionale chiara: Mantenere la logica condizionale semplice e chiara. Evitare condizioni troppo complesse o nidificate eccessivamente. Se necessario, suddividere le condizioni complesse in funzioni separate per migliorare la leggibilità e la manutenibilità.**

- **Prevenzione di errori comuni: Ad esempio, prima di eseguire operazioni che richiedono specifici requisiti (come la divisione), verificare sempre che tali requisiti siano soddisfatti (ad esempio, che il denominatore non sia zero).**

cicli e iteratori

In PHP, i cicli sono strumenti fondamentali che permettono di ripetere l'esecuzione di blocchi di codice fino a quando una condizione specificata non viene soddisfatta.

Ci sono diversi tipi di cicli in PHP, ognuno con le sue caratteristiche e casi d'uso.

Ecco una panoramica dei cicli più comuni in PHP.

Iteratori in php

Tipi di Cicli in PHP

1. Ciclo while: Il ciclo while esegue un blocco di codice finché la condizione specificata rimane vera. È utile quando il numero di iterazioni non è noto prima dell'inizio del ciclo.
2. Ciclo do-while: Simile al ciclo while, ma con una differenza chiave: il blocco di codice viene eseguito almeno una volta prima che la condizione venga testata, garantendo così che il codice all'interno del ciclo venga eseguito almeno una volta.
3. Ciclo for: Il ciclo for è utilizzato quando il numero di iterazioni è noto. Richiede tre espressioni: l'inizializzazione, la condizione e l'incremento, tutti specificati all'interno della dichiarazione del ciclo.
4. Ciclo foreach: Il ciclo foreach è ottimizzato per iterare attraverso gli elementi di un array o di un oggetto. È il modo più semplice e diretto per attraversare un array in PHP.

Esempi di Cicli in PHP

Esempio 1: While

```
1. <?php
2. $i = 0;
3. while ($i < 5) {
4.     echo "Il valore di i è: $i<br>";
5.     $i++;
6. }
7. ?>
```

Esempi di Condizioni in PHP

Esempio 2: do - While

```
1. <?php
2. $i = 0;
3. do {
4.     echo "Il valore di i è: $i<br>";
5.     $i++;
6. } while ($i < 5);
7. ?>
```

Esempi di Cicli in PHP

Esempio 3: For

```
1. <?php
2. for ($i = 0; $i < 5; $i++) {
3.     echo "Il valore di i è: $i<br>";
4. }
5. ?>
```

Esempi di Cicli in PHP

Esempio 4: Foreach

```
1. <?php
2. $colori = array("rosso", "verde", "blu");
3. 
4. foreach ($colori as $colore) {
5.     echo "Colore: $colore<br>";
6. }
7. ?>
8. 
```

Teoria degli Array in PHP

Gli array in PHP sono strutture dati che possono contenere più valori, i quali possono essere accessibili tramite chiavi numeriche o stringhe.

Gli array sono estremamente versatili e possono essere utilizzati per una vasta gamma di scopi, come la memorizzazione di dati, la gestione di liste, configurazioni, e molto altro. Di seguito, esploreremo la teoria degli array in PHP e alcune pratiche comuni.

Teoria degli Array in PHP

Tipi di Array:

1. **Array indicizzati numericamente: Gli indici sono numeri interi. Questi array possono essere creati automaticamente quando si aggiungono valori senza specificare una chiave.**
2. **Array associativi: Gli indici sono stringhe. Questo tipo di array è utile per rappresentare strutture più complesse, come le proprietà di un oggetto.**
3. **Array multidimensionali: Gli array che contengono altri array come valori. Sono utili per rappresentare tabelle, matrici, o qualsiasi set di dati annidati.**

Teoria degli Array in PHP

Operazioni comuni sugli Array:

- **Inserimento: Aggiungere elementi a un array.**
- **Rimozione: Rimuovere elementi da un array.**
- **Accesso: Accedere a un elemento tramite la sua chiave.**
- **Traversa: Passare attraverso tutti gli elementi di un array.**
- **Ordinamento: Cambiare l'ordine degli elementi in un array.**

Teoria degli Array in PHP

Funzioni Utili in PHP per Gli Array:

- **array_push(): Aggiunge uno o più elementi alla fine di un array.**
- **unset(): Rimuove un elemento da un array.**
- **array_pop(): Estrae l'ultimo elemento di un array.**
- **sort(): Ordina un array.**
- **array_merge(): Combina due o più array.**
- **count(): Conta il numero di elementi in un array.**
- **foreach(): Itera su ogni elemento di un array.**

Teoria degli Array in PHP

```php
<?php
// Creazione di un array indicizzato numericamente
$numeri = array(1, 2, 3, 4, 5);
echo "Numero iniziale: " . $numeri[0] . "\n"; // Output: 1

// Aggiungere elementi
array_push($numeri, 6);
echo "Aggiunto un elemento: " . end($numeri) . "\n"; // Output: 6

// Rimozione di un elemento
unset($numeri[1]); // Rimuove il valore in posizione 1 (2)
echo "Elemento rimosso, nuovo array: ";
print_r($numeri);

// Creazione di un array associativo
$persona = array("nome" => "Mario", "età" => 30);
echo "Nome: " . $persona["nome"] . "\n"; // Output: Mario

// Traversa di un array con foreach
foreach ($numeri as $indice => $valore) {
    echo "Indice: $indice, Valore: $valore\n";
}

// Ordinamento di un array
sort($numeri);
echo "Array ordinato: ";
print_r($numeri);

// Combina due array
$combinato = array_merge($numeri, $persona);
echo "Array combinato: ";
print_r($combinato);
?>
```

Esempi di Cicli in PHP

Ciclo for con Condizioni Multiple

Il ciclo for è estremamente versatile e utile per eseguire un numero predefinito di iterazioni.

Nell'esempio dato, il ciclo for itera da 1 a 100, e dentro il ciclo, la condizione if verifica se l'indice corrente è divisibile sia per 3 sia per 5.

Questo approccio è comune nei problemi di programmazione dove si devono eseguire azioni specifiche a intervalli regolari o sotto condizioni specifiche.

È un ottimo esempio di come le operazioni matematiche possono essere integrate nei cicli per filtrare o manipolare dati.

Esempi di Cicli in PHP

Ciclo for con Condizioni Multiple
Questo esempio utilizza un ciclo for per calcolare la somma dei numeri interi da 1 a 100 che sono divisibili sia per 3 sia per 5.

```
1. <?php
2. $somma = 0;
3. for ($i = 1; $i <= 100; $i++) {
4.     if ($i % 3 == 0 && $i % 5 == 0) {
5.         $somma += $i;
6.     }
7. }
8. echo "La somma dei numeri divisibili per 3 e 5 tra 1 e 100 è: $somma";
9. ?>
```

Esempi di Cicli in PHP

Ciclo while con Controllo di Flusso Interno

Il ciclo while è ideale per situazioni in cui il numero di iterazioni non è noto prima dell'esecuzione del ciclo, poiché continua ad eseguire il blocco di codice finché la condizione specificata è vera.

Nell'esempio mostrato, il ciclo while elabora gli elementi di un array fino a quando non incontra un valore negativo, dimostrando come gestire input dinamici o incerti efficacemente. Questo tipo di ciclo è comune in scenari di analisi o elaborazione di dati sequenziali dove il punto di terminazione può variare.

Esempi di Cicli in PHP

Ciclo while con Controllo di Flusso Interno
Questo esempio mostra come un ciclo while può essere utilizzato per leggere e processare dati fino a quando non si incontra una condizione specifica, in questo caso, un valore negativo.

```
1. <?php
2. $dati = [10, 20, 30, -1, 40, 50];  // Array di esempio
3. $index = 0;
4. $somma = 0;
5.
6. while ($index < count($dati) && $dati[$index] >= 0) {
7.     $somma += $dati[$index];
8.     $index++;
9. }
10. echo "La somma fino al primo valore negativo è: $somma";
11. ?>
```

Esempi di Cicli in PHP

foreach per Navigare e Modificare Array Multidimensionali

Il ciclo foreach è particolarmente adatto per la manipolazione di array in PHP, poiché permette di iterare facilmente su ogni elemento dell'array.

Nell'esempio, foreach è utilizzato per attraversare e modificare un array multidimensionale, incrementando ogni elemento di uno.

Questo ciclo semplifica la lettura e la scrittura in strutture di dati complesse, rendendo il codice più pulito e meno incline a errori di indice fuori limite.

Esempi di Cicli in PHP

foreach per Navigare e Modificare Array Multidimensionali
Questo esempio utilizza un ciclo foreach per iterare attraverso un array multidimensionale, modificando gli elementi di ogni sotto-array.

```php
<?php
$matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
];

foreach ($matrix as $key => $array) {
    foreach ($array as $subkey => $value) {
        // Incrementiamo ogni valore di 1
        $matrix[$key][$subkey] = $value + 1;
    }
}

print_r($matrix);
?>
```

Esempi di Cicli in PHP

Combinazione di Cicli e Condizioni

Questo esempio dimostra l'efficacia di combinare cicli e condizioni per eseguire operazioni più complesse su array.

Utilizzando il ciclo foreach con una condizione if interna, si selezionano e modificano solo certi elementi dell'array (in questo caso, raddoppiando i numeri pari).

Questo pattern è estremamente utile per filtrare dati e applicare trasformazioni condizionali, una tecnica comune in molti aspetti dello sviluppo software, dalla manipolazione di dati alla generazione di output personalizzati.

Esempi di Cicli in PHP

Combinazione di Cicli e Condizioni
Un esempio di come i cicli possono essere combinati con le condizioni per eseguire compiti più complessi, come filtrare e operare su elementi specifici di un array.

```php
<?php
$numeri = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
$filtrati = [];

foreach ($numeri as $numero) {
    if ($numero % 2 == 0) {
        array_push($filtrati, $numero * 2);
    }
}
echo "Numeri pari raddoppiati: ";
print_r($filtrati);
?>
```

in PHP

Funzioni

PHP offre una vasta gamma di funzioni incorporate che coprono un'ampia varietà di funzionalità, consentendo agli sviluppatori di eseguire compiti comuni come la manipolazione di stringhe e array, l'elaborazione di file, la gestione delle sessioni e molto altro.

Queste funzioni standard sono parte integrante del linguaggio e sono ottimizzate per offrire prestazioni elevate e una sintassi semplificata, riducendo così il tempo di sviluppo e il codice necessario.

Funzioni In php

Passaggio di Parametri:

- **Per Valore: Di default, i parametri vengono passati per valore, il che significa che qualsiasi modifica al parametro all'interno della funzione non influenzerà il valore esterno.**

- **Per Riferimento: Passando un parametro per riferimento, le modifiche al parametro all'interno della funzione influenzeranno il valore originale.**

Funzioni In php

Definizione di una Funzione: Una funzione in PHP è definita usando la parola chiave function, seguita da un nome univoco per la funzione, una coppia di parentesi (che possono contenere parametri opzionali) e un blocco di codice racchiuso tra parentesi graffe.

Sintassi Base:

```
1. function nomeFunzione() {
2.     // Codice da eseguire
3. }
```

Funzioni In php

Parametri della Funzione: Le funzioni possono avere parametri, che agiscono come variabili locali all'interno della funzione. I parametri sono specificati tra le parentesi nella definizione della funzione e sono separati da virgole.

Esempio con Parametri:

```
1. function saluta($nome) {
2.     echo "Ciao, " . $nome . "!";
3. }
4.
5. saluta(“Mirko”)
```

Funzioni In php

Esempio con Valore di Ritorno: Per eseguire una funziona che riporta andiamo a vedere come usare la key-word “return”

```
1. function somma($a, $b) {
2.     return $a + $b;
3. }
```

Esempio di Chiamata:

```
1. saluta("Mario");
2. $risultato = somma(5, 3);
3. echo $risultato;  // Stampa 8
```

Funzioni In php

return nella Funzione: Una funzione può restituire un valore al chiamante, che può essere di qualsiasi tipo, utilizzando la parola chiave return. Se return non viene specificato, la funzione restituirà NULL di default.

Molteplici valori di ritorno: Per restituire più valori, puoi usare un array.

```
1. function calcola($a, $b) {
2.     $somma = $a + $b;
3.     $moltiplicazione = $a * $b;
4.     return array($somma, $moltiplicazione);
5. }
```

Funzioni In php

Definire i Parametri Default: Puoi definire valori predefiniti per i parametri per rendere la funzione più flessibile.

```
1. function saluta($nome = "Ospite") {
2.     echo "Ciao, $nome!";
3. }
```

Funzioni In php

Spiegazioni Tecniche

- **Ambito delle Variabili (Scope): Le variabili definite all'interno di una funzione sono locali alla funzione stessa e non possono essere accessibili al di fuori di essa, a meno che non siano restituite o non siano definite come globali.**

- **Riutilizzo del Codice: Le funzioni permettono di riutilizzare il codice senza doverlo riscrivere, migliorando l'efficienza dello sviluppo.**

- **Manutenibilità: Le funzioni rendono il codice più facile da testare e mantenere, poiché ogni funzione è progettata per eseguire una specifica attività.**

Funzioni In php

Crea un ciclo che una alla volta stampi 4 funzioni coi loro risultati tenendosi i dati e sommandoli alla fine:

Somma, divisione, moltiplicazione, sottrazione

La somma di tutti i risultati va stampata e anche i singoli risultati

Funzioni Incorporate

Le funzioni incorporate di PHP sono progettate per essere facilmente accessibili e richiedono spesso solo una chiamata con i parametri necessari.

Ad esempio, per la manipolazione delle stringhe, PHP offre funzioni come strlen(), str_replace(), e strpos().

Per le operazioni su array, funzioni come array_merge(), sort(), e count() sono disponibili e pronte all'uso.

```
1. $text = "Hello World";
2. echo strlen($text); // Output: 11, lunghezza della stringa
```

Funzioni Definite dall'Utente

PHP permette anche agli sviluppatori di definire le proprie funzioni. Questo è particolarmente utile per incapsulare blocchi di codice che vengono riutilizzati, mantenendo così il codice organizzato e più facile da mantenere.

Una funzione può essere definita utilizzando la parola chiave function, seguita da un nome univoco, un elenco di parametri tra parentesi e un blocco di codice racchiuso tra parentesi graffe.

```
1. function maxNum($num1, $num2) {
2.     if ($num1 > $num2) {
3.         return $num1;
4.     } else {
5.         return $num2;
6.     }}
7. echo maxNum(4, 7); // Output: 7
```

Esempi Pratici

Esempio 1: Funzione per Invertire una Stringa

```
function invertiStringa($stringa) {
    return strrev($stringa);
}

echo invertiStringa("hello");  // Stampa "olleh"
```

Esempi Pratici

Esempio 2: Funzione per Verificare se un Numero è Pari o Dispari

```
function isPari($numero) {
    return $numero % 2 == 0;
}

if (isPari(4)) {
    echo "Il numero è pari.";
} else {
    echo "Il numero è dispari.";
}
```

Esempi Pratici

Esempio 3: Funzione che Restituisce Array

```
function creaArray($elemento1, $elemento2) {
    return array($elemento1, $elemento2);
}

$result = creaArray("mela", "banana");
print_r($result);  // Stampa Array ( [0] => mela [1] => banana )
```

Scrivi una funzione in PHP che calcola e restituisce la media dei numeri contenuti in un array.

Requisiti

- **La funzione deve prendere un array di numeri come input.**
- **La funzione deve restituire la media dei numeri nell'array.**
- **Gestisci il caso in cui l'array sia vuoto restituendo zero.**

Scrivi una funzione in PHP che trova e restituisce il valore massimo in un array di numeri.

Requisiti

- **La funzione deve prendere un array di numeri come input.**
- **La funzione deve restituire il valore massimo trovato nell'array.**
- **Gestisci il caso in cui l'array sia vuoto restituendo null.**

Scrivi una funzione in PHP che calcola la potenza di un numero dato una base e un esponente. La funzione deve calcolare il risultato senza utilizzare la funzione integrata pow() per l'elevamento a potenza.

Requisiti

- **La funzione deve accettare due parametri: la base e l'esponente.**
- **La funzione deve restituire il risultato dell'elevamento a potenza.**
- **Assicurati di gestire correttamente il caso in cui l'esponente sia zero (ogni numero elevato alla potenza di zero è 1).**

Teoria degli Oggetti in PHP

PHP è un linguaggio di programmazione ampiamente usato per lo sviluppo web, che supporta la programmazione orientata agli oggetti (OOP).

La programmazione OOP in PHP consente agli sviluppatori di organizzare il loro codice in modo più modulare e riutilizzabile.

Ecco una panoramica delle caratteristiche principali degli oggetti in PHP, seguita da esempi pratici per iniziare.

Teoria degli Oggetti in PHP

Le classi sono i fondamenti della programmazione orientata agli oggetti. Una classe agisce come un modello per gli oggetti e definisce le caratteristiche (proprietà) e le azioni (metodi) che quegli oggetti possono avere.

Un oggetto è quindi una specifica istanza di una classe, creata in memoria con dati reali. Pensare alle classi come ricette e agli oggetti come i piatti preparati usando quelle ricette può aiutare a comprendere questo concetto.

Ogni oggetto ha uno stato indipendente, e le interazioni tra oggetti differenti avvengono attraverso i loro metodi. La definizione della classe include un metodo speciale chiamato costruttore, che viene eseguito automaticamente ogni volta che viene creata una nuova istanza della classe, permettendo l'inizializzazione degli oggetti.

Teoria degli Oggetti in PHP

Creare e Usare un Oggetto
Diamo un'occhiata a come creare e utilizzare un oggetto in PHP partendo dalla classe Automobile definita sopra.

```
// Creazione di un'istanza della classe Automobile
$auto = new Automobile("Fiat", "500");

// Accesso al metodo dell'oggetto
echo $auto->dettagliAuto();   // Output: Marca: Fiat, Modello: 500
```

Teoria degli Oggetti in PHP

Una classe in PHP è un template per gli oggetti, e un oggetto è un'istanza di una classe. Le classi possono contenere proprietà (variabili) e metodi (funzioni).

```php
class Automobile {
    public $marca;
    public $modello;

    public function __construct($marca, $modello) {
        $this->marca = $marca;
        $this->modello = $modello;
    }

    public function dettagliAuto() {
        return "Marca: " . $this->marca . ", Modello: " . $this->modello;
    }
}
```

Teoria degli Oggetti in PHP

Incapsulamento

L'incapsulamento è il concetto di nascondere i dettagli interni della classe e di esporre solo ciò che è necessario.

In PHP, questo è realizzato usando modificatori di accesso come public, private, e protected.

- **public: La proprietà o il metodo è accessibile da ovunque.**
- **private: La proprietà o il metodo è accessibile solo all'interno della classe stessa.**
- **protected: La proprietà o il metodo è accessibile all'interno della classe e dalle sue sottoclassi.**

Teoria degli Oggetti in PHP

```php
class Account {
    private $saldo;

    public function __construct($saldoIniziale) {
        $this->saldo = $saldoIniziale;
    }

    public function deposita($importo) {
        if ($importo > 0) {
            $this->saldo += $importo;
        }
    }

    public function preleva($importo) {
        if ($importo <= $this->saldo) {
            $this->saldo -= $importo;
            return $importo;
        }
        return 0; // Non sufficiente saldo
    }

    public function getSaldo() {
        return $this->saldo;
    }
}
```

Teoria degli Oggetti in PHP

Ereditarietà

L'ereditarietà permette a una classe di ereditare proprietà e metodi da un'altra classe. In PHP, usiamo la parola chiave extends per indicare l'ereditarietà.

```
1. class Veicolo {
2.     public $ruote = 4;
3. }
4.
5. class Moto extends Veicolo {
6.     public function numeroDiRuote() {
7.         return $this->ruote;
8.     }
9. }
```

Teoria degli Oggetti in PHP

Un metodo magico è un metodo di una classe che viene chiamato automaticamente al verificarsi determinati eventi all'interno dell'oggetto istanziato della classe.

Durante la presenziane della classi, ci siamo già imbattuti nel metodo costruttore, che non è altro che un metodo di un classe chiamato automaticamente non appena viene istanziato un oggetto della classe stessa. Lo rivediamo rapidamente oggi, assieme agli altri metodi magici:

- **__construct**
- **__destruct**
- **__toString**
- **__get**
- **__set**
- **__call**
- **__callStatic**

Teoria degli Oggetti in PHP

Polimorfismo

Il polimorfismo è la capacità di trattare oggetti di classi diverse attraverso un'interfaccia comune. In PHP, il polimorfismo può essere implementato tramite interfacce o classi astratte.

```
interface Animale {
    public function suono();
}

class Cane implements Animale {
    public function suono() {
        return "Bau bau";
    }
}

class Gatto implements Animale {
    public function suono() {
        return "Miao";
    }
}
```

Teoria degli Oggetti in PHP

Errori Comuni nella OOP con PHP

1. **Mancato uso di $this all'interno dei metodi**
   - **$this si riferisce all'istanza corrente della classe. Dimenticarsi di usarlo all'interno di un metodo per accedere alle proprietà o ai metodi dell'oggetto può portare a errori.**
2. **Visibilità impropria delle proprietà e dei metodi**
   - **Impostare tutti i metodi e le proprietà su public può esporre troppo l'interno della classe, violando il principio di incapsulamento.**
3. **Dipendenze circolari tra classi**
   - **Quando due o più classi sono dipendenti l'una dall'altra, può portare a problemi di manutenzione e a difficoltà nell'identificare gli errori.**
4. **Mancato riuso del codice tramite composizione**
   - **Preferire l'ereditarietà alla composizione può portare a gerarchie di classi complesse e difficili da gestire.**
5. **Overriding improprio di metodi**
   - **Sovrascrivere un metodo di una classe genitore senza mantenere una firma coerente o senza chiamare il metodo originale quando necessario (usando parent::methodName()).**

Teoria degli Oggetti in PHP

La comprensione approfondita degli oggetti in PHP passa attraverso la conoscenza di diversi concetti avanzati e pratiche che migliorano la progettazione, l'efficienza e la manutenibilità del codice.

Ecco alcuni elementi aggiuntivi e specifiche d'uso che possono aiutarti a sfruttare al meglio la programmazione orientata agli oggetti in PHP.

Teoria degli Oggetti in PHP

Quando usiamo OOP in PHP, considera questi aspetti:

- **Manutenzione e Scalabilità: Utilizza OOP per rendere il codice più organizzato, manutenibile e scalabile.**
- **Riuso del Codice: Sfrutta ereditarietà, composizione e traits per massimizzare il riutilizzo del codice.**
- **Principi SOLID: Segui i principi SOLID per una progettazione software robusta e manutenibile.**

Teoria degli Oggetti in PHP

Le interfacce definiscono un contratto che le classi possono implementare. Esse specificano quali metodi una classe deve implementare, senza definirne l'implementazione. Le interfacce sono utili per garantire che diverse classi condividano la stessa interfaccia di metodi, pur avendo implementazioni diverse.

```
interface Animale {
    public function emettiSuono();
}

class Cane implements Animale {
    public function emettiSuono() {
        echo "Bau";
    }
}

class Gatto implements Animale {
    public function emettiSuono() {
        echo "Miao";
    }
}
```

Teoria degli Oggetti in PHP

Classi Astratte
Le classi astratte sono classi che non possono essere istanziate direttamente e sono destinate a essere sottoclassi. Una classe astratta può contenere metodi astratti con nessuna implementazione, obbligando le classi figlie a fornire implementazioni specifiche.

```
abstract class Animale {
    public $nome;
    abstract public function emettiSuono();

    public function setNome($nome) {
        $this->nome = $nome;
    }
}

class Cane extends Animale {
    public function emettiSuono() {
        echo "Bau";
    }
}
```

Teoria degli Oggetti in PHP

Traits
I traits sono un meccanismo per il riutilizzo del codice in linguaggi che supportano l'ereditarietà singola, come PHP. Un trait è simile a una classe, ma è destinato a raggruppare funzionalità in modo flessibile. Un trait può essere usato in diverse classi per offrire capacità aggiuntive senza forzare una relazione di ereditarietà.

```
1. trait Loggable {
2.     public function log($msg) {
3.         echo "Log: $msg";
4.     }
5. }
6.
7. class File {
8.     use Loggable;
9. }
10.
11. class User {
12.     use Loggable;
13. }
14.
```

Teoria degli Oggetti in PHP

Namespaces
I namespaces sono un modo per incapsulare elementi come classi, interfacce, funzioni e costanti. Sono particolarmente utili in applicazioni di grandi dimensioni per evitare conflitti di nomi e organizzare meglio il codice.

```
1. namespace MyProject\User;
2.
3. class User {
4.     // Implementazione della classe User
5. }
6.
7. namespace MyProject\Util;
8.
9. class User {
10.     // Implementazione della classe User diversa
11. }
12.
```

Teoria degli Oggetti in PHP

Gestione delle Eccezioni

La gestione delle eccezioni in PHP permette di catturare e gestire errori in modo controllato. Le eccezioni possono essere lanciate (throw) e catturate (catch) per prevenire errori durante l'esecuzione e gestire condizioni di errore in modo più pulito e organizzato.

```
class User {
    public function setName($name) {
        if (empty($name)) {
            throw new \InvalidArgumentException("Il nome non può essere vuoto.");
        }
        $this->name = $name;
    }
}
try {
    $user = new User();
    $user->setName("");
} catch (\InvalidArgumentException $e) {
    echo "Errore: " . $e->getMessage();
}
```

Esercizio di PHP con Condizioni

Creeremo una classe chiamata Cinema che può tenere traccia dei film disponibili e delle prenotazioni. Implementeremo metodi per aggiungere un film, prenotare un posto e visualizzare le prenotazioni.

Struttura della Classe

1. **Classe Cinema:**
   - **Proprietà:**
     - **film: Un array associativo dove la chiave è il titolo del film e il valore è il numero di posti disponibili.**
   - **Metodi(funzion):**
     - **aggiungiFilm($titolo, $posti): Aggiunge un nuovo film con i posti disponibili specificati.**
     - **prenotaPosto($titolo): Prenota un posto per il film specificato se ci sono posti disponibili.**
     - **mostraPrenotazioni(): Visualizza tutti i film e i posti rimanenti.**

Esercizio di PHP con Condizioni

Creeremo una classe chiamata ContoBancario che permette di gestire il saldo di un conto, permettendo operazioni di deposito e prelievo, oltre a fornire il saldo attuale del conto.
Struttura della Classe

1. **Classe ContoBancario:**
   - **Proprietà:**
     - **saldo: Un valore float che tiene traccia del saldo corrente del conto.**
   - **Metodi:**
     - **__construct($saldoIniziale): Costruttore che inizializza il conto con un saldo iniziale.**
     - **deposita($importo): Aggiunge l'importo specificato al saldo del conto.**
     - **preleva($importo): Sottrae l'importo specificato dal saldo del conto, se il saldo è sufficiente.**
     - **getSaldo(): Restituisce il saldo corrente del conto.**

Esercizio di PHP con Condizioni

Creeremo una classe chiamata Inventario che permette di gestire un elenco di articoli, ognuno rappresentato da una classe Articolo. Ogni articolo avrà un nome e una quantità in magazzino. Implementeremo metodi per aggiungere articoli all'inventario, rimuoverli e aggiornare la loro quantità.

Struttura delle Classi

1. **Classe Articolo:**
   - **Proprietà:**
     - **nome: Il nome dell'articolo.**
     - **quantita: La quantità dell'articolo in magazzino.**
   - **Metodi:**
     - **__construct($nome, $quantita): Costruttore che inizializza il nome e la quantità dell'articolo.**
2. **Classe Inventario:**
   - **Proprietà:**
     - **articoli: Un array che memorizza gli oggetti Articolo.**
   - **Metodi:**
     - **aggiungiArticolo($articolo): Aggiunge un nuovo articolo all'inventario.**
     - **rimuoviArticolo($nome): Rimuove un articolo dall'inventario per nome.**
     - **aggiornaQuantita($nome, $quantita): Aggiorna la quantità di un articolo esistente.**

Esercizio di PHP con Condizioni

Definiremo una classe chiamata Macchina che rappresenta una macchina operativa in una fabbrica. Ogni macchina può essere attivata o disattivata, e può produrre un articolo quando è attiva.

Struttura della Classe

1. **Classe Macchina:**
   - **Proprietà:**
     - **stato: Un booleano che indica se la macchina è attiva (true) o disattivata (false).**
   - **Metodi:**
     - **attiva(): Attiva la macchina.**
     - **disattiva(): Disattiva la macchina.**
     - **produci(): Produce un articolo se la macchina è attiva, altrimenti stampa un messaggio di errore.**