From novice to complete projects

Developing Apps With PHP Smarty And MySQL

A book by

SHOUVIK SARKAR

DEVELOPING APPS WITH PHP SMARTY AND MYSQL

From novice to complete projects

BY **SHOUVIK SARKAR**

PUBLISHED BY

Lulu Press, Inc.

Morrisville, North Carolina, USA.

First Published In World Wide In 2026 By Lulu Press Inc.

ISBN : 978-1-312-19962-0

Lulu Press, Inc.
Morrisville, North Carolina, USA.
https://www.lulu.com/

ACKNOWLEDGEMENTS –

For this book I would like to acknowledge my fellow readers for which this work is dedicated to. They had been the focus behind writing this book. Besides I would also like to support people around me who had worked as a dedicated support to me for writing this book.

Besides most thanks goes to the publishers without whom this and my previous works could always stayed as a fiction only. So I am thankful to all of them.

PREFACE -

This book is a book related to computer programming language PHP which will explain from basics of coding to writing high level and complex projects. This book is for anyone who want to master PHP be him yet to come into the field of programming or have good hold of the language. They will get a good idea on how to build complex projects altogether using PHP besides Smarty templating engine and MYSQL as database.

This would be helpful for its readers to get a good practical knowledge on how to set up your environment, write complex codes for modules and integrate modules together by keeping it simple. If you are totally new into coding still you can find this useful and learn how to write good code and create complex projects by not getting off the hold of it.

Thanks and I hope all readers of my book would like my work and this book would interest to as many people as it can.

INDEX

INTRODUCTION

Lets start our journey into coding using PHP, Smarty, MySQL along with using REST APIs. In the beginning we shall go through what actually these technologies are and what they offer us to develop awesome market ready complex apps. I believe you have a basic ideas of what front end, back end and database is? If not still its fine as I am going to explain them too in the beginning. The purpose of this book is to make even a novice who is just making entry into software development can be able to develop complex apps quite efficiently. So lets start our discussion into it.

A web based app has majorly three functional components - a front end, a back end and a database to store user data for showing records specific to the user. Here for front end we shall use Smarty and PHP, for backend it will be PHP while the database we will be using is MySQL. So lets look into it. Here for backend we are referring to an API which is a smart move for developing scalable apps. This approach helps in modular micro service based architecture which is very important for scaling the features of the application as we go.

But besides this the most important aspect needed while creating a web app is a server. For PHP apps we generally use the Apache server. So lets dive in brief into the technologies that is PHP, Smarty and MySQL and also understand the basics of front end, back end and database in brief. So let’s start with PHP.

What is PHP?

In simple terms PHP is a server side scripting language which is used to return a response from the server after an activity or request that is placed. Lets dive a little deeper into it. For PHP the current version is 8.x and is a very much advanced version than it’s previous ones. This is a highly secure language using which we can develop commercial apps with ease. Though few frameworks are too being developed for PHP but using the core version gives much more flexibility and we shall mostly discuss about the core features here. We shall dive into the programming features in later chapters but I believe this has given a good idea of what actually PHP is. We can use it for both frontend logic and backend APIs. So lets give a look into Smarty.

What is Smarty?

Smarty in most general terms is a templating engine which can be used alongside PHP to write front end pages having specific syntax for logical cases. So how it works? Basically there is a library for it using which we can simply pass data to the template file of our choice to display. The most required feature of Smarty is we can implement new front end without much affecting the business logic by just passing the data to new

template files. We shall look more into it and its coding format in later parts. Now lets move on brief a bit of MySQL.

What is MySQL?

MySQL is a database or a software that stores data. It runs as a standalone server and external apps can connect it to the port to send commands and store data. It is very flexible and secure which uses mostly two database engines namely InnoDB an MyISAM for storing data. Data is stored within separate databases by creating different tables which makes it very structured. PHP can connect to the server and communicate using specific commands for storing or retrieving data. The language used to operate on the stored data is called SQL or Structured Query Language.

So combining these above technologies we can create full fledged commercial applications with ease. But now let's dive a bit into the architecture of commercial applications and how we can make them secure and scalable. To make scalable apps as discussed earlier the most important thing is building a modular architecture. It makes it easier to maintain and integrate. So we shall look into that too later.

But how can we make such a development environment easily so that it can resemble a production environment? We shall look into that now. For that first we need to be familiar of the concept of virtual hosts. Virtual hosts allow us to host multiple websites through a single server. This can be created in development environment too and mapped with domain by setting up in hosts file to be working in local server. This is beneficial for creating multiple projects or an API and frontend both in same server and carry on with the development.

First of all lets get some idea of what code to be present in a virtual host to create a separate domain and website. I am writing the code below. Lets check it first and understand it.

```
<VirtualHost 127.0.0.1:80>
        DocumentRoot [Root_Directory]
        ServerName [Domain.TLD]
</VirtualHost>
```

For creating a virtual host for the port 80 which is a HTTP port and not an HTTPS we do not require to provide the SSL details. So the constituents of this are the document root where the website files will be hosted along the domain which will redirect it to the IP 127.0.0.1 at port 80 and will serve the files in document root after getting interpreted by the PHP interpreter.

```
<VirtualHost 127.0.0.1:443>
        DocumentRoot [Root_Directory]
        ServerName [Domain.TLD]
        SSLEngine on
        SSLCertificateFile [SSL_CERT_FILE]
        SSLCertificateKeyFile [SSL_KEY_FILE]
</VirtualHost>
```

For the port 443 which is a HTTPS port we need to provide the SSL certificate details which includes the key and certificate along with the same as mentioned above. Also to be noted in this we need to enable SSL first by setting the SSLEngine to on. The requests to the domain will be redirected to port 443 if https is used in url and the constituents at the document root will be served after getting interpreted by PHP followed by encrypting using the SSL files. To be noted the virtual hosts file is located under Apache followed by conf and extra directories respectively by default. This is an overview of how Apache works at top level. Now lets look into some more concepts.

Next thing is to make this virtual hosts work we need to edit the hosts file and update it with the hostname along with the IP address. The location file location is different for Windows and Linux system but easily can be found it in the OS directory. So once we setup this much we are ready to write some code. So before writing and testing any line we need to be ensured that both Apache and MySQL servers are up and running.

For creating a very simple test file we can simply write Hello World in an editor and save it as index.php file inside the document root specified. Be reminded the default file that is processed by Apache inside the document root is the one named index. The extension can be html, htm, php etc. This is specified by the settings of the Apache. So now opening the domain.tld will display a blank page with Hello World written on it. Hence we executed our very first php file and tested it in a browser. Now we shall look into some of the things that we would require while coding and debugging the application.

One of the most important aspects of a PHP and Apache setup is the .htaccess file. This is used to route the requests from url to specific files in the directory. This is very important to create urls without direct access to files and also customize url and directory patterns. We shall look further into it later. The smarty uses a specific extension for template files which is .tpl. The smarty compiler compiles these files to php and stores in a compiled directory inside the template directory. It also adds the benefits of caching as the files are compiled only when it is changed. So first request after a change takes some time but later requests are faster as the compilation does not takes place then. Besides these xml or env files are good option to store the variables

that are specific to the whole site. These are some among the many other concepts that makes developing scalable apps using php easier.

For the part of API development we shall use mostly REST API's specifically using two methods which is GET and POST. These are mostly used methods for API and for accessing the APIs from PHP frontend curl is one of the best inbuilt library provided. This is easy, secure and one the most efficient library to send the requests and process the response from the API. Creating an API and a microservice based architecture is very useful in maintaining the application and scale it at a later stage. So let's now move on to see a basic prototype on how we can create scalable architecture and the app code maintainable by using separate design files and separate business logic. So let's look into that.

As mentioned the best way to develop an app is keep design files and business logic separate. And also as we discussed smarty is one of the best tools to make it happen. The design files are kept in a separate theme folder which are called from an intermediary php file using smarty class. This intermediary file are the entry point to any page as it firstly authenticates the request followed by fetching the data from a common file which has functions to get the data and passing this data to the smarty templates. Some of the files need not be accessed from browser and hence can be restricted using htaccess file for the site and includes files like config and site constants file. The rest of the files like ajax or other php files in the application can be prevented direct access from url using a check of a constant value of a variable at the top of the file. This is a very basic idea to create secure and scalable applications in php and smarty.

For API part there is a very secure library developed by me itself named SARK Auth which can be very useful for securing the API using OAuth 2.0. This can secure the API along with maintaining a good code structure for processing request and responses along with getting data or perform the required operations in a separate file than the one where requests are processed is always a good practice. So these are some good practices for creating APIs.

Now lets move on to the frontend part. We shall be using Smarty template files which generally have an extension of .tpl. These files uses its own smarty syntax for displaying data or perform any operation on the data. We shall look more into the various syntax for performing operation over these files later in this book. So far what we have seen are the basic ingredients which needs to be ready in order to cook the food which is the application. So we shall look into each of them in detail later in this book. So now lets move on to see a basic software that can be very useful for setting up email server in local host. The software is named as hmailserver. So lets look into it now.

Hmailserver can be downloaded from their official website. Once downloaded it needs to be installed and then set up for localhost. For detailed steps you can check their official documentation. Multiple domains can be setup over it and for each domain multiple email accounts can be too which can be useful for testing apps. This provides a brief idea on how to use the application. Once set up the emails created shall work for localhost. In fact it is only an advanced SMTP server with enhanced feature sets.

Now lets move back into the coding part comprising of PHP, Smarty and MySQL. Lets look into what actually coding in PHP looks like in brief. In brief PHP is high level object oriented programming language which though is not a compiled rather interpreted. PHP language is more of a scripting type. Its features constitutes of things like multiple data types, variables, constants, functions, classes, conditional blocks and many other object oriented features. Now lets look into MySQL. MySQL using SQL as its language and having various database engines to store data is a structured way to store data and retrieving from there. It is very compact, robust and secure and hence is used widely. Various modules in MySQL are its ability to create procedure, triggers and events. Now lets look briefly in smarty. In smarty it has its own type of syntax which are always written inside curly braces. It has its own form of writing conditional statements or using php functions. This is in brief how the codes in php, smarty or mysql are written.

I shall also like to brief you a bit about how its about writing php code for complex projects. Though mostly there are lesser amount of ready made packages available writing complex projects may need to write a lots of code but to work on such projects keeping the code structure modular always helps. For connection to database there are inbuilt functions provided by php while always an api and microservice based architecture is beneficial. We shall look more in detail into this later.

Now lets move on to next chapter and dive deeper into these technologies and further into how we can use them to develop complex commercial applications and much more.

BASICS OF PHP

PHP is a scripting and interpreted programming language as we already discussed and is mostly used for developing web based apps. The web server for PHP is Apache and the complete package having all the basic requisites for PHP development are available in Xampp which is freely available. This can be downloaded from their official website and installed to use it and develop apps in PHP. For more details and downloading Xampp the below website can be used.

Official Website of XAMPP: https://www.apachefriends.org/

After downloading Xampp from the above website it can be installed. The present version of PHP is 8.x. The latest Xampp version too comes along with this latest version of PHP. Hence using it we can use the exquisite and powerful features of PHP 8.x. The MySql version in Xampp is 5.x. To use latest MySql you can download that from their official website and use it by running the server at port 3306 without starting the one packaged with Xampp. The website to download MySql is given below.

Official Website of MySQL: https://www.mysql.com/

For accessing the MySQL usually MySQL Workbench or a PHP application named PHP My Admin can be used. Now before digging more into software and set up now lets brief a bit on to how to write codes in PHP which this chapter is more of intended to. So in this part we shall look into what actually comprises the PHP programming language. Lets begin into the tour.

To start the tour first lets look into the data types that are available in PHP. The various data types supported by PHP are listed here –

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

To be noted is that while declaring a variable a datatype is not required and a value can be only assigned by specifying the variable name to it. PHP internally assigns the variable a datatype. Also we can change its datatype simply by changing the value of the variable to of some other datatype. This makes working with variables in PHP very simple and easy as datatypes are not strictly bounded to variables in PHP.

Now lets go through the various datatypes available in PHP one by one to have a better idea of all of them. The first datatype mentioned is String. A string is a sort of datatype in which characters or a group of characters which in general terms words or sentences can be stored. A string must be declared within quotes to distinguishing it as a string. The set of character can be any of the keyboard input enclosed within couple of quotes. To specify the same quote used to define the string inside it a backslash can be used. Specifically, within two quotes whatever is specified is said to be a string and the variable to which it is assigned will get the datatype of a String. Let's see an example below.

```
$str = "Hello";          // String
```

We use $ symbol to specify a variable. We shall look into that later. But for the purpose of datatypes this is how a String is defined. So now lets look into the next datatype in the list which is Integer. An Integer is a sort of datatype which can hold any numeric value. Be noted the value can be positive or negative though can not be a decimal number. Also there is a range for integers which depends on the architecture of the CPU but we shall discuss that later and the range is generally very high. Now let's see how to declare an Integer variable.

```
$i = 100;          // Integer
```

In the above example there are no quotes and this is how integers are declared in PHP. The variable i will be created with the datatype of integer. Next in the list of datatypes is Float which differs from Integers narrowly. Let's look into that now. Float is generally used to declare decimal numbers. Float has a greater range than Integer and can be used to declare decimal numbers. Let's see how they are declared.

```
$num = 10.6;          // Float
```

Floats are also declared without quotes but when a decimal number is assigned to a variable the variable created will automatically have the datatype Float. Now let's move on and see about Booleans. Booleans are type of datatype that can specifically hold two values 0 or 1 more specifically true or false. Now let's look into it in more details.

Boolean are basically a type of datatype which can hold true or false values. The value can also be mentioned in integer as 0 or 1 where 1 resembles true while 0 resembles false. This is used basically in implementing logical conditions in a program. To see how it is represented let's see below.

```
$switch = true          // Boolean
```

So as we looked in brief into the Boolean datatype we can assume that it is a very important part in formulating any program. Let's move on to Arrays. Arrays are very important in storing and passing data. Arrays itself are basically of three types – indexed, associative and multidimensional. For storing and passing data arrays are more important part. Lets check how we declare arrays.

```
$array = array();         // One dimensional array
$mutid_array = [][]     // Multi dimensional array
```

Muti dimensional arrays are used to store more complex data while single dimensional arrays can store sets of data in a structured manner. Hence arrays are used to pass data between functions, pages or even APIs as it provides a structured way to do the same. Now lets move on to the next datatype in PHP which is a object. An object is in fact an instance of a class which is used to access properties and methods of a class. Lets see how an object is declared.

```
$obj = new Class();
```

An object is in fact created by instantiating a class by using the new keyword. While instantiating we can pass various parameters to the class which will be directly passed to the constructor of the class. We shall look more into classes and objects later. For the moment lets look into the null datatype. A null datatype is a sort of datatype which is yet not assigned or logically do not hold any of the values. We declare null datatype using the null keyword which is displayed below.

```
$a = null;
```

Null is different from empty or blank values in the sense that null values are indeed no value at all. Null is a reserved keyword in PHP and cannot be used to name functions. So while we look more of its usages later lets look into next datatype resource.

Resources are sort of datatypes that holds link to external entities of PHP code like a file / directory pointer, a database connection, socket handles etc. These are created when a request to such a external resource is made and the variable holds the pointer to the resource for future processing. An example of a resource can be like below.

```
$fp = fopen('file', w+);
```

In the above code $fp holds a link to the external file and performs operation as described to it on the file through itself. So these are all the datatypes that php supports. Next we shall look into all the access levels / scopes that are provided in PHP.

In PHP there are generally three access modifiers. They are listed below.

- Public
- Protected
- Private

Besides these three two more access modifiers can said to exist. They are default which is same as public and other is global. We shall look into all of these access modifiers one after another. Let's start with Public.

A variable when declared with public modifier then it can be accessed from within the class, classes that inherit or even outside the class. Generally if we declare a variable within a block then it can be used within that or its inner block level only. That is a general rule for using variables which we shall look later. Also we should note that if we do not specify any access modifier to a variable by default it is assumed to be public. So now let's move on to the next modifier which is Protected.

A variable when declared with protected modifier then it means it can be used only from within the class or from classes that inherit it. The variable is not accessible from outside the class. This is how data can be protected and it's value not be modified from places it is not intended to. Let's see now about the Private modifier and where its value is limited to.

A variable declared with private modifier can be accessed only from within the class and hence has a local scope. Private variables are used to carry on values for variables that are required only inside the class for various permutations. Generally public functions are declared where they are used and then the function returns a value which are used by external objects for different requirements.

Generally, any function declared outside the class level are global in nature and can be accessed from throughout the code. But inside any class or function they are directly not available rather needs to be passed as parameters. Global variables are declared using the global keyword. Generally, by default also any variable declared out any of the blocks acts as global variables.

So far we have looked into variables only. Now let's look into what constants are in PHP, how they are declared and accessed. Constants are generally defined in PHP using define function or the constant keyword. A constant can be declared to have value of any of the data types but for constants once they are declared their value can not be modified throughout the execution of the code. If the value of any constant is modified or redeclared an error message is shown by the PHP compiler. Let's see below how a constant is declared in PHP.

```
define('NAME', 'VALUE');

const NAME = 'VALUE';
```

So basically these are the two ways by which constants are declared in PHP. We shall look into more of their uses later. Now let's move on and look into the building blocks of PHP language which include Namespaces, Classes, Functions and Traits. After this we shall look into the OOPs (Object Oriented Programming) concepts that PHP supports. So let's start with Namespaces.

Namespaces are generally used to differentiate and organize code so as the code can be better readable and used in a more efficient way. Mostly namespaces are a great way to use classes of same name but different usages by enclosing them in different namespaces. Let's look into the syntax of a namespace before going a bit deeper into it's basics. To declare a namespace we simply use the below syntax.

```
namespace PROJECT_NAME\NAMESPACE_NAME
```

To call the namespace we can use two ways. First is by the use keyword as below.

```
use PROJECT_NAME\NAMESPACE_NAME
```

Then we can call any classes between the namespace by simply creating the object of the class. In an alternative way we can call the class directly as below.

```
$obj = new \PROJECT_NAME\NAMESPACE_NAME\CLASS_NAME
```

This is followed by we can use the object of the class as usual. As we have already seen the basic needs and characteristics of a namespace let's know about another important property of namespace before moving on to classes. This is that a namespace must be declared at the top of the file and while we include the use statement any code written below it will automatically be considered within the namespace. By keeping this in mind let's move on to classes.

Classes in object oriented programming are very good procedure to segregate the whole code into meaningful and structured manner. We can divide the whole code into different modules and functionalities ach of which can be maintained by different classes to maintain code readability, structure and also it enhances security and good code. We shall look into the various OOPs concepts like inheritance, polymorphism etc later. For now lets look how classes are used in PHP. For PHP also it is no different from other languages. Objects can be created of the class and its members be variables or functions can be accessed using the object based on its modifiers. Lets see how a class is called in PHP.

```
$obj = new CLASS_NAME

$obj -> CLASS_METHOD / CLASS_VARIABLE
```

This is how methods in a class are accessed. For static methods or variables a double colon operator can also be used to directly access the member without actually creating an object of the class.

```
CLASS_NAME::STATIC_METHOD
```

So this is what classes provide PHP and make it a very good and versatile language. Now let's move on and look into functions in PHP. In PHP also functions can have a return type or it only perform specific function only. More over functions can be declared inside a class or even outside it which would have a global scope by default. Functions can take parameters as input or they can be without any parameters too. The parameters can be initialized with a default value in which case they need not be passed when the function is called. They have access modifiers if they are declared within a class. Besides that they can return a value after performing some operations which can be assigned directly to a variable for further operations. Functions are indeed a great way to divide the code in blocks for better maintainability. PHP supports function overloading or overriding which we shall look later. Lets see how functions are declared now inside a class.

```
ACCESS_MODIFIER function FUNCTION_NAME(OPTIONAL_PARAMETER){

}
```

This is the general way how a function is declared. Now as we checked PHP supports single inheritance only, but to equip PHP with the benefits of multiple inheritance too traits are introduced in PHP. But before we move on and take a look into traits lets look into the basic concepts of object oriented programming which are Inheritance, Overloading, Overriding and Abstract which are supported in PHP. Lets start with inheritance.

Inheritance is the ability to use the variables and functions of the parent class that it inherits without explicitly creating an object as the members are of the same class. This property of OOPs provides the ability to PHP to create sub classes. Specifically though PHP supports single inheritance only. It means a class in PHP can not inherit multiple classes. Following is the syntax used to inherit classes.

```
class CHILD_CLASS extends PARENT_CLASS {

}
```

By using the above syntax child_class can use the members of the parent_class. Also the construct of the parent_class can be initialized using below code.

```
parent::__construct();
```

So this is how single inheritance is implemented in PHP. We shall dive deeper into it later. Lets move on and check how function overloading works in PHP. In fact in PHP function overloading works very differently. The two basic points to note is that function overloading is performed by the magic method __call() and the number of parameters that can be passed is always dynamic. Below is the syntax for function overloading in PHP.

```
function __call($name_of_function, $arguments){

}
```

Here the magic method __call should always be public while for specific function we need to write various logical conditions and process the arguments accordingly. This is the general way how function overloading is performed in PHP. While for the static instance of the method overloading we can use the __callStatic() method. Here is how this is used.

```
function __callStatic($name_of_static_function, $arguments){

}
```

Both works in similar way except how they are accessed. Below is how they are accessed in context of general and static methods only after creating objects of the class. For example if both the methods are inside class Test and an object $test is created for the class then they can be accessed as below.

```
$test -> $name_of_function($arguments…)          // Default overloaded function

$test :: $name_of_static_function($arguments…)   // Static overloaded function
```

Only other point here to note is that $arguments are passed as comma separated when called whereas they are accessed inside the respective __call function as an array. So this is how overloaded functions are used in PHP. Now move on and let's look into function overriding in PHP.

Method overriding in PHP is simple and is similar to other languages. We can redeclare the same function in the class in which it is inherited and change how it acts. To note in method overriding both function name and number of parameters must be

same. Here is how this is used in PHP. For example in one class named Parent a method is declared name func() with parameters $par. Then in the Child class which inherits the parent class the overriding can be performed as below.

```
$func($par){

    // New body of the function

}
```

By using the above approach when we call the function from parent and child classes we get different values from it. So this is how function overriding works in PHP.

Now let's move on how abstract classes are used in PHP. Abstract classes are classes in which only the functions and the whole class is declared but not implemented. Though general functions can be placed inside them, whereas to declare abstract function the abstract keyword is used. Similarly to declare the abstract class too abstract function is used. The major point to note is no object can be created for the class. Generally other classes inherit them and the abstract functions are redeclared in them to be used by objects of that class. Lets see how abstract classes are declared.

```
abstract class ABSTRACT_CLASS {

    abstract function ABSTRACT_FUNCTION() { /* empty function */ }

}
```

So this is how abstract classes and functions are used. Lets move on and get back into look how traits are used in PHP. Traits are exclusive concept in PHP that enables a PHP programmer to reuse functions in multiple classes. Traits can have abstract methods too. Below is how traits are declared.

```
trait TRAIT_NAME {

    function FUNCTION_NAME(){ /* function body */ }

}
```

To use the trait in multiple classes they must be called using the use keyword in each class. After that we can use the functions defined in the trait. Lets see how the syntax of using a trait is.

```
class CLASS_NAME {
```

```
    use TRAIT_NAME;
}
```

After calling the trait the function inside the trait can be accessed using the object of the class. So these are the various concepts that OOPs supports and are also can be used for writing programs in PHP. So as we complete the OOPs concepts in PHP now we shall move forward and look into what conditional statements and loops are supported in PHP next.

So to begin the section lets start with the conditional statements. The most important conditional statements used in PHP are if-else and switch statements. The if-else statements are itself of multiple types. So lets start with the if-else statement only.

The if-else statements are of three variations. The syntax of each are mentioned below.

- **if block :**

```
if <condition> {
    /* code block */
}
```

- **if-else block :**

```
if <condition> {
    /* code block */
} else {
    /* code block */
}
```

- **if-elseif-else block :**

```
if <condition> {
    /* code block */
} else if <condition> {
```

```
    /* code block */
}
 else {
    /* code block */
}
```

As suggested by syntax these are three types of blocks that are available as a part of if statements. In first one if the conditions evaluates to true the block will be executed. In the second part if the if condition is true then the block executes otherwise the else block will be executes. For the last part there are multiple if conditions and the topmost if condition that evaluates to true the respective block will be executed. So this is how the if and if-else conditions are available in PHP. So now lets move on and look into the switch statements that are available in PHP.

Switch statements or switch cases are another way to execute conditional statements in PHP. A switch statement can have multiple branches or cases. Each case has a condition which when evaluates to true then that corresponding block gets executed. Below is the syntax of the switch case code.

```
switch <expression> {
    case <value-one>:
        // code block
        break; // if no break statement is used then next values are also checked
    case <value-two>:
        // code block
        break;
    default:
        // code block - if no value matches expression then this block is executed
}
```

According to the above syntax switch case can be explained as follow. The expression is the one that is evaluated which returns some value. If the return value matches any of the ones mentioned in case then the corresponding block code will be executed. To be noted is the break statement without which the next cases are also checked. Also there is a default case which is executed when none of the cases is matched. This is how a switch case is programmed. Also these are the only two conditional statements that PHP supports. So now lets move on to the loops that PHP supports.

PHP supports two loops basically which too has two variants. They are while and for loops in which while has while or do-while loops. Again for is having for or for-each loops. Lets look into each one by one and start by while loop. In while loop if a condition returns true then the code block will be executed till the condition returns false. Each time the block is executed the condition is rechecked and till the condition returns true it keeps on executing. Lets see the syntax.

```
while <condition> {

    // code block

}
```

To keep note on two statements continue and break. The continue statement passes to the loop and again check the condition by keeping in view the code below the continue statement is not executed. On the other hand break statement exits the code from the loop into the next sequence of codes. This is how while loop works. Next lets move on the do-while loop. The syntax of do-while loop is below.

```
do {

    // code block

} while <condition>
```

In the do-while loop the code block inside the do part is executed by default and after it only the condition is checked. Rest of the execution of the do-while loop is similar to while loop. So in do-while loop the first time code block is executed and after it only the conditions are checked for further execution.

So lets look now into the for and for-each loop. In for loop we initialize a variable, check it that an expression returns true and change the value of the variable so that the condition returns different values on each loop. Below is the syntax of for loop.

```
for (<variable_initialize>; <check_condition>; <increment_or_decrement_value>) {

    // code block

}
```

In the above loop an variable say $i is initialized by a value say 0. Then the condition is executed say $i < 5. According to this till the variable $i is less than 5 the loop is executed. After it the value of the variable $i can be incremented or decremented as required. Say here we increment hence write $i++. So the value of $i will increase in each loop and till its less than 5 the code will be executed. Once the condition returns false the loop is executed. To note there can be two types of increment or decrement namely pre-increment and post-increment which are written as ++$i and $i++. This we shall see in detail later. Also we can use continue or break statements in for loop which functions as described earlier. So this is how for loop acts and now lets look into for-each loop.

A for-each loop iterates through each element of an array and various operations can be performed on the element as per iteration. Lets look below into the syntax of a for-each loop.

```
foreach (<array> as <element>){

    // code block

}
```

According to the syntax if the array is empty the code block inside loop is not executed. In other cases every element in the array will be iterated. There is a variation in the for-each loop in which the keys and values of each element is iterated. So this is all to complete the loops section. Now lets move on and get an idea of various operators that are available in PHP.

The list of operators in PHP is long. We shall go through most of them in brief. Lets start with the arithmetic operators. Below is an expression that denotes how we can use them. After it we shall mention each operator and why we use them for further understanding.

```
<RESULT> = <VARIABLE_ONE> <OPERATOR> <VARIABLE_TWO>
```

Here to note is that = is also an operator named assignment operator. Also to note that in place of variables we can pass constants or static values too. Now let’s look into the list of mathematical operators and their usages.

Addition (+) Multiplication (*) Modulus (%)

Subtraction (-) Division (/) Exponentiation (**)

So these are all the arithmetic / mathematical operators available in PHP. On applying this the result is assigned to the variable left hand side of the assignment operator. Now lets move on and check the assignment operators itself.

For assignment operators we shall firstly look how they are used followed by what are various assignment operators in PHP that can be used. Below is how an assignment operator is used.

<RESULT> <OPERATOR> <OPERATION>

Here to note is that the operation can be simple including two operands or even complex with multiple operands with arithmetic, string, logical or any other operators or simply an value. Below is the list of assignment operators and why they are used.

= (Assignment) += (Add and Assign) -=(Subtract and Assign)

*= (Multiply and Assign) /= (Divide and Assign) %= (Modulus and Assign)

**= (Exponent and Assign)

So after going through the list of assignment operators now lets look into very important part which is the comparison operators. These operators are used for multiple conditional requirement which provides the base of any logic. They always returns true or false in result. Lets firstly look how a comparison operator is used.

<VARIABLE_ONE> <OPERATOR> <VARIABLE_TWO> // true / false

So here also the variables can be replaced by constants or static values. To be noted is any comparison operator values that must be true or false. Lets look now into the various comparison operators.

== (Equal) === (Equal with Datatype) != (Not Equal)

<> (Not equal) !== (Not Equal with Datatype) < (Less than)

> (Greater than) <= (Less than or Equal) >= (Greater than or Equal)

<=> (Returns -1 [less], 0 [equal] or 1 [greater])

These operators can be used over arrays as well with similar functionalities. So as we looked into the various comparison operators in PHP, now is the time to move on logical operators which are very important in writing any logical code in PHP. PHP has four logical operators which are namely and, or, xor and not. Lets see how they are used and their short hand for writing them in code. Logical operators are very useful in formulating various logic and their operation are based on logic requirements of the program. Lets see how they are used.

<CONDITION_ONE> <OPERATOR> <CONDITION_TWO> // true / false

Here mostly a condition is some expression that returns true or false. Hence operands for mostly logical operators are true or false. If in case they are not Boolean, any value that is empty or zero resembles a false whereas other values resembles true. To note that the result of logical operators are also Boolean that is true or false and mostly used in conditional blocks. Now lets look into various logical operators in PHP.

AND / && (Logical AND Operation) OR / || (Logical OR Operation)

XOR (Logical XOR Operation) ! (Logical NOT Operation)

So these are the four logical operators available in PHP. Besides these useful operators there are some operators that can be used with string datatype only and used for various string modification functions. Lets see them.

String operators are generally used for string concatenation or string concatenating assignment operators. They are used to make sentences from words which are dynamic or many other operations as and when required. Lets see how they are used. The resultant is also always a string.

<RESULT> = <STRING_ONE> <OPERATOR> <STRING_TWO>

Now lets look the various operators available in string operations.

. (Concatenation) .= (Concatenate and Assign)

So these are all the operators that are available in PHP and we shall look later into some of their usages with examples. Now lets look into another type of expression cum operator which are called ternary operators. Ternary operators are used to write short hand expressions for if statements which makes the code tidy. Lets look into some of the ternary operators.

First of them is the shorthand if-else operator and below is mentioned how it is used. Lets give a look into it.

<EXPRESSION> ? <BLOCK_ONE> : <BLOCK_TWO>

Here if the expression returns to true then block_one will be executed otherwise block_two will be executed. So this is similar to the conventional if-else block. Now lets move on to check the called null coalescing operator. This is used as below.

<RESULT> = <VALUE_ONE> ?? <VALUE_TWO>

Here if value_one is set and not null then value_one will be assigned to result otherwise value_two. So they are short hand and hence makes the code smaller. This is how various operators are available in PHP and how they are used. Now to carry on our discussion lets look into the next feature of PHP that is called try-catch which is used for handling runtime errors in PHP.

A try-catch is a special block which in case of any runtime errors catches the exception and in that specific exception segment its handling is done. There can be multiple catch blocks with a try block and they are executed serially in which the error is catch in the topmost matching block. There are specific exceptions or a common general exception which can catch any of the errors. Lets see how a try-catch block is written by default.

```
try {

    // code block

} catch (<specific_exception> $se) {

    // catch and process a specific exception

} catch (exception $e) {

    // catch and process any exception

}
```

So in this block as we see specific exceptions segment are written above the default exception block and such errors are processed otherwise on the reverse case such exception will itself get catch and covered by default exception segment. In the catch blocks in case of any runtime errors the error can be processed as per requirement. But one point to note here is as a part of good coding practices try-catch blocks should be used for errors that could not be handled by logic in any way. Such errors generally resembles database connection lost, api response error from curl or similar cases. We

shall look more into this later. Now lets move on to see some of the special powers that PHP possess. And to start the part lets begin with super global variables.

Super global variables are those special variables in PHP which are inbuilt and are used for fetching some special data. To see which are the super global variables lets enlist them firstly. Below is the list of super global variables.

- $GLOBALS
- $_SERVER
- $_GET
- $_POST
- $_REQUEST
- $_COOKIE
- $_FILES
- $_ENV
- $_SESSION

So these are the list of super global variables in PHP. $GLOBALS is used to access any global variable at any place inside a code like class or function. Hence any global variable $v can be accesses directly from any place in code using $GLOBAL['v']. To note the specific property of all super global variables are that they can be accessed from anywhere in code. Now lets look into the other ones.

$_SERVER is used to get any of the server specific functionalities of a page like self address, origin address etc. It can also get if the request is get or post to use proper super global variable later on. Except it can get the complete request url, user agent or even if url is over https or not. There are specific keys to get each of these values. We shall look some of them later. Now lets move on to the next one.

$_GET is used to get any parameter from the URL by just mentioning the key. For example any URL can be as mentioned below.

<protocol>://<domain.tld>/?<parameter_one>=<value_one>&<parameter_two>=<value_two>

Here any of the values of the parameters can be accessed directly using the key name with the $_GET variable as $_GET[<variable_one>]. Generally $_GET is used to access parameters which are publicly visible as that is the only possibility in $_GET requests. Now lets look into $_POST requests.

$_POST is similar to $_GET except parameters and their values are passed in a secure manner. This means they are not visible in URL. The variables are accessed similarly using the key name by which they are passed.

$_COOKIE is used to access the cookies stored in clients browser. They can be accessed using the cooking name to get the value of the cookie. We shall look further into cookies later and the importance of storing cookies encrypted every time. To access a cookie value we can access using its key as $_COOKIE[<cookie_name>] directly.

$_REQUEST is a special super global which can get the values of any of the three types which are $_GET, $_POST or $_COOKIE. The value can be accessed by its key. So if we do not know the request method we can directly use $_REQUEST but it is not a good practice as in this case any intruder can simulate $_POST requests by sending $_GET to create trouble or have some other issues too.

$_FILES is a super global to access files send over the network. The page handling the request can access the files using this super global variable. It provides the size, name, type, temp_name of the file sent. Generally files can be sent by form submit using "multipart/form-data" or through curl requests. So to access the file this can be used and for further processing like uploading to server etc. To upload to server a method is available in PHP called move_uploaded_files(<$_FILE['{name}']['temp_name']>, <path_on_server>). So this is used to upload files. We shall look more of this later.

$_ENV is a super global which can be used to access data set by the function putenv("<variable_name>=<variable_value>"). So to get the variable set in the env $_ENV["<variable_name>"] can be used. Generally site specific variables can be set using the method and later accessed using $_ENV from any part of the code.

The last in the list but one of the most important one is the session variables. This is used after the session_start() function and used until session_destroy() is called. In between any function stored in $_SESSION variable can be access using the same from any part of the application not necessarily the variable defined be in the code line. For example a variable defined by $_SESSION["<variable_one>"] can be accessed from other part using same $_SESSION["<variable_one>"] name until a session destroy is called in the application. Session variable has time limits though after which it is also automatically destroyed. Any specific session variable can be unset using session_unset(<variable_name>) function. Sessions are generally used to store user specific details for short time to show them user specific data and forms a very important part of almost any major application.

So these are all of the super global variables which are very useful in PHP. Now lets move on to the next part which are magic functions. So lets begin the discussion on

magic functions. Magic functions are some of the major inbuilt functions in PHP and they generally starts with a double underscore (__). Now lets look at the various magic functions in PHP and why they are used. We shall list some of the major ones and study others later as per requirements.

- __construct(<parameter_list>)
- __destruct()
- __call(<function_name>, <parameter_array>)
- __serialize()
- __unserialize(<DATA>)

So each of them are used differently. __construct() method is a method which is directly called when a new instance of the class is created. We can pass parameters to the class and they can be accessed from the __construct() method. Though here the parameters are optional. Lets see the syntax below. Firstly lets assume we had this class.

```
class <CLASS_NAME> {

    function __construct(<PARAMETER_LIST>){

        // process the parameters along with any other logic

    }

    function __destruct() {

       // logic while class object is destroyed

    }

    function __serialize() {

       // return array format data that will automatically be serialized

    }

    function __unserialize(<ARRAY_DATA>) {

       // gets the array format data from object passed

    }
```

```
    function __call(<FUNCTION_NAME>, <PARAMETER_ARRAY>) {
        if (<FUNCTION_NAME> == <FUNCTION_A>) {
            // process parameters passed with <FUNCTION_A>
        }
        …
        …
        if (<FUNCTION_NAME> == <FUNCTION_B>) {
            // process parameters passed with <FUNCTION_B>
        }
    }
}
<OBJECT> = new CLASS_NAME(<PARAMETER_LIST>);
```

So here we have declared the class and created an object <OBJECT> from it. So lets look how each of the magic methods that are mentioned can be used. Firstly the __construct() method will automatically be called while we create the object. Destruct is automatically called when we destroy the object of the class. This can be done using unset() function calling the object or even automatically when the script finishes execution. __call() function is called whenever a method is called using the object. It checks for an logic inside the __call() method for the method name. In case it finds it performs the respective logic using the parameters passed. Mostly __call() method is used for function overloading in PHP. __serialize() function is used for serializing the data. This is accessed by serialize(<OBJECT>) method and this returns the serialized data of the array that is returned from __serialize() method within the class. For the __unserialize() method we use the unserialize(<SERAILIZED_DATA>) method and on which the __unserialize method retrieves the data in unserialized array format. The array can then be processed further.

So these are how and why some of the magic methods are used. We shall further of these or other magic methods later and as per requirements. Now lets move on and go through the last part of this chapter specifying some of the major inbuilt functions that are available in PHP.

PHP is a very powerful programming language providing many inbuilt functions for different tasks. We shall look into some of those major inbuilt functions which are very useful and how they are used. This extends from databases, api calls, file uploads, sessions, file handling and few others for datetime, regular expressions etc. So lets continue our discussion and start with mysql functions which is the most extensively used language with PHP.

MySQL is very much connected with PHP and the developers provide an extensive range of functions for connection, data handling and disconnection. Lets list the major mysql functions available in PHP.

- mysqli_connect(<host>, <user>, <password>, <db>, <port>)
- mysqli_close(<connection>)
- mysqli_query(<connection>, <query>)
- mysqli_insert_id(<connection>)
- mysqli_fetch_array(<result>, <fetchtype>)
- mysqli_num_rows(<result>)

Besides these functions MySQL a bunch lot of functions but these are the most important among them and we shall discuss these serially. The first call in case of mysql queries is the mysqli_connect() call. In this call various parameters passed are database hostname, username, password, name and port respectively. This returns the database connection object which can be passed to the other functions according to requirements.

To close the database connection we use the function mysqli_close() and pass the connection object to it. This closes the connection and hence prevents multiple concurrent connections to the database which can exceed the max connection limit of the MySQL server otherwise. Hence whenever a query is executed the close connection call is a very good practice to follow.

After this the most important function being used is mysqli_query(). In this function the connection object and the SQL query are passed respectively. The SQL query here can be any of the queries including Insert, Update, Select, Delete or some other. For any Data Manipulation Language (DML) query this statement returns true or false in case of successful or unsuccessful operations respectively. For Data Query Language (DQL) query this returns the result object. In case of a new row created in a table we can get the primary key of that row using the function mysqli_insert_id() and passing the connection object into it only.

Next function mostly used in PHP is mysqli_fetch_array() which is used in case of DQL queries. As it returns a result object, the same is passed to the function followed by fetch type which can be MYSQLI_BOTH, MYSQLI_ASSOC or MYSQLI_NUM. This

returns the respective type of array from the result. This array can be further processed to retrieve various individual data.

One more function we shall discuss here is the mysqli_num_rows() which too takes the result object as parameter. It returns the number of rows retrieved in the result object which is also useful for many cases.

There are many more mysql functions but we shall look into that later on as per requirement. One more thing to note is that as a good coding practice every mysql function should be used inside a try-catch block. The major reason is that any error in the MySQL server side can not be handled in PHP and hence if occurs can be handled by the try-catch block. By keeping this note in mind lets move on to see some other useful functions in PHP.

The next important such function used for making API calls is the curl function. This is used for making external API calls be it GET or POST or some other. Lets see the syntax of a curl function.

```
<curl_identifier> = curl_init();

curl_setopt(<curl_identifier>, CURLOPT_URL, <remote_url>);

curl_setopt(<curl_identifier>, CURLOPT_HTTPHEADER, <header>);

curl_setopt(<curl_identifier>, CURLOPT_POST, <true/false>);

curl_setopt(<curl_identifier>, CURLOPT_POSTFIELDS, <data>);          // optional

<response> = curl_exec(<curl_identifier>);

if (curl_errno(<curl_identifier>)){ // handle error }

curl_close(<curl_identifier>);
```

So these are the major fields used with curl though there are many other options available with curl_setopt() which too can be used. Firstly curl is initialized which is followed by setting URL, Header, POST or GET and Data. After this the curl_exec() function calls the API and returns the response which can be set to a variable for further processing. Lastly we must close the connection using curl_close() so that the limit of maximum connections is not reached. We can use another option using curl_setopt() which is CURLOPT_RETURNTRANSFER. It accepts two values true or false which returns the header or not in the response. So this is a very basic curl usage. We shall check more detailed with examples in subsequent chapters.

The next important in the lot is file handling. For file handling data must be submitted through POST from forms. GET methods does not support file handling in native form rather data can be sent using blob only for GET requests. In native scenario a form that accepts multipart/form-data can only send file over to the server from the client browser. Once the file is uploaded and form is submitted this needs to be handled in the server. We shall look into that here. More detailed file upload operations we shall look later. In the server side the files are checked using the $_FILES super global variable. Using this variable we can get file name, size, extension and tmp_name. To upload the file tmp_name is used while the other parameters can be used for validating and further processing the files. Lets see a basic file upload operation in the server side.

```
<remote_file> = $_FILES[<identifier>][‘tmp_name’];

<file_name> = $_FILES[<identifier >][‘name’];

<file_size> = $_FILES[<identifier >][‘size’];

<file_error> = $_FILES[<identifier >][‘error’];

if (!<file_error> and <file_size> <= <maximum_allowed_size>){

    <status> = move_uploaded_file(<remote_file>, <destination_path>);

    if (<status>){ // file uploaded successfully } else { // upload failed }

}
```

This is a very basic file upload script in PHP. Please note the major function to copy the file from temp location to destination path which is move_uploaded_file(). This accepts temporary file and destination path as parameters respectively. We shall look into its more detailed usages later on. For now its good to know this part only. Now lets move on to the next function which are session handling functions.

Session handling is a very important part in web applications to display user based content For the same some sessions are created for different users and some user specific data are stored based on which respective data are displayed to the user. The same is done by a session id for each user which is stored in the user’s browser on which all data are stored and fetched for the user. Lets see how a session is coded in PHP. Lets see the syntax below.

```
session_start();

$_SESSION[<variable_name>] = <value>;        // can be user_id
```

So as we see on calling the function session_start() the session id is generated and session is created on the server based on the session id. Now using $_SESSION super global variable various data is stored for the session in the server. So these session variables can be fetched at any time from any page from the session data stored in the server based on session id. To note session_start() must be called and only once before accessing the session variables. Now to end the session session_destroy() is used which destroys the session cookie from browser and hence session from the server. Except this to unset specific variables from session data session_unset() can also be used. Lets check the syntax of destroying the session.

session_destroy();

This function ends the session for the user by removing the browser cookie from client and session data from server. This is how a session works. Now lets see how to set required cookie data in client browser. This is very simple. Cookies are set using setcookie() method while they can be accessed using $_COOKIE super global. To unset a cookie manually we just need to changed the parameter valid_time of the cookie to any past value using the same setcookie() method. Lets see the syntax below.

setcookie(<name>, <value>, <valid_time>, <domain>);

Here the parameters are cookie name which is used to access the cookie, cookie value which the value set for the cookie, valid time which is till when the cookie is valid (this is the actual time in seconds the cookie is valid) and the subdomain or complete domain the cookie can be used. To unset the cookie the valid time just needs to be changed to any past time using the same function. Now to access the cookie in server side we can use $_COOKIE as below.

<cookie_variable> = $_COOKIE[<cookie_name>];

Using this line we can retrieve the cookie from the browser. So this is how session and cookies works in PHP.

Now lets see how a file is accessed and processed in PHP. Below is the syntax of reading a file.

<file_pointer> = fopen(<file_path>, <mode>);

// process the file

// fgets(<file_pointer>) | fwrite(<file_pointer>, <text>)

fclose(<file_pointer>);

There are different modes for opening a file which can be read, write, append or a combination of them. To process the file based on requirement they can be read a line or a character and assign the content to a variable in PHP. Similarly writing can be done to the file. There are different functions available in PHP for the same. For example fgets() reads a line in PHP while fwrite() writes to the file. We shall look into them in detail in later part of this book. For now please take a note that there are many simple functions also available to process files some of which are listed below.

- file_get_contents(<file_path>); // Read the whole file
- file_put_contents(<file_path>. <mode>); // Write to the file
- unlink(<file_path>); // Deletes the file

Besides these there are many other functions too in PHP to process files which we shall look later as per requirements. For now lets move on and give a brief look into some other important functions that are used in PHP. Mostly we shall look here into dat and time functions and regular expressions.

So lets start with date and time functions. We enlist here the major ones with procedural style used for date time processing.

- date(<format>, <timestamp>);
- time(); // returns the UNIX timestamp
- date_default_timezone_set(<timezone>);
- strtotime(<datetime_string>); // Also accepts special parameters

So these are some of the major date and time functions available in PHP. Here in date() function the format can be passed which are special string representations each denoting different format and the timestamp which would be converted. The time() function gives the UNIX timestamp for now starting 01/01/1970. The date_default_timezone_set() sets the timezone which is passed as a variable while strtotime() takes the date and time string and converts that into date time object. It can also take special parameter strings like “+2 days”, “+1 month” which sets the variable to the mentioned parameter from now by adding or subtracting to present time. So these are some of the date and time functions while many other we shall check later as per requirements.

Lets now see how regular expressions are processed in PHP. Regular expressions are special format strings which can be checked if valid or not using the preg_match() function in PHP. This has variety of uses like validating email, phone numbers, password formats etc. So lets see how a preg_match() is written in PHP.

<regext_variable> = /<regex_pattern>/; // Special patterns are for different cases

<response> = preg_match(<regex_variable>, <string_to_check>);

So here the pattern is defined firstly. Then the pattern is checked using the preg_match() function passing the defined pattern and string to check as its parameters. The function returns true or false based on the match found or not respectively.

The next function preg_match_all() returns all the matches to the pattern. This is written as below.

preg_match_all(<regex_variable>, <string_to_check>, <matches>);

This function accepts a third parameter matches which returns an array with all the matches corresponding to the regex_variable in the string_to_check. So these are some of the important functions used to process regular expressions in PHP. We should note one more function which is simple and useful to check for emails or urls.

filter_var(<string_to_check>, <filter_name>);

Here the filter_name can be varied to either check the string or sanitize the string according to needs. Some of the values for filter_name is mentioned below.

<response> = filter_var(<string_to_check>, <check_filter_name>); // true or false

Some values for check_filter_name can be “FILTER_VALIDATE_EMAIL”, “FILTER_VALIDATE_URL”, “FILTER_VALIDATE_INT” or few others. While for sanitizing filter name this can be used as below.

<response_string> = filter_var(<string_to_check>, <sanitize_filter_name>); // string

Here the response from the function is a processed string. Some such sanitize filter is “FILTER_SANITIZE_STRING”, “FILTER_SANITIZE_EMAIL” etc. So this how filter_var() can also be used to process input and validate the same.

So for concluding this chapter we have looked into most of the functionalities and features supported by PHP. For now this is good to continue while we shall look and go through other things that are required in the course of the study. PHP is quite long and have many features which can easily found over the internet. In this book I am just looking to get you go on and get an idea on how to build complex and important applications which solve real world problems. For the purpose this is good to know in Basics Of PHP while as stated we shall look into anything else we require during the course of the book. So on this note lets end this chapter and move on to look into the Basics of MySQL which is also very important for creating real world applications.

BASICS OF MYSQL

MySQL is one of the most robust, versatile, secure and easy to use database engines available. As it is packaged with Xampp it has been an integral part for development with PHP. But I would suggest to download the latest version from their official website mentioned in Chapter 1 as the packaged one is an older version of it. So upon download it can be installed by following the steps in the installer. Once installed and server started it runs on the 3306 port by default. So to access the database server I would suggest using PHPMyAdmin or MSQL Workbench as these are the better IDE available for MySQL. A CLI tool is though by default available. So once the setup is done we shall see here how we can work with MySQL including with PHP for developing data based applications. So lets start our discussion.

Though lets discuss brief the major things that we are going to discuss in this chapter. This includes database engines, user management, DDL (Data Definition Language), DQL (Data Query Language) and DML (Data Manipulation Language) queries, transactions, triggers, procedures and events, keys, supported data types, using MySQL in PHP and some of the best practices to be used during using MySQL. So lets continue and start our discussion with the database engines. Before proceeding also note that for IDE choice we shall refer in this chapter mostly according to PHPMyAdmin.

So we shall now start our discussion and firstly go through how a database software like MySQL work actually and afterwards the various engines available in MySQL. In basic level a MySQL seems to work using SQL queries which are used to query the database tables for DDL, DQL or DML queries. But actually whenever a database is created in MySQL a directory is created for same which constitutes of each table that it contains or created within it. These files are different for different storage engines but they exists with separate folders for separate databases. Except this there are some common files ibd and iblog which consists of temporary database structure and latest queries in buffer. These files are mostly used for crash recovery. All these various files stores data in specific format. The whole processing of commands to write to these files along with reading from these files to send the data in tabular format to output is done by the MySQL server. The MySQL server also consists of the SQL parser to understand the SQL language, most extensively used database language which helps to perform respective functions. So this is a bit of internal functioning of MySQL. Now we shall look into the various storage engines available in MySQL.

Generally MySQL has multiple engines but we shall focus mostly on two engines – MyISAM and InnoDB. Lets start the discussion with MyISAM. Some of the most common and important features of MyISAM engine is it is faster for reads, portable, though support concurrent inserts it is slower for DML queries, do not support transactions, not support foreign keys and not have sufficient crash recovery mechanisms. This has some implications like the database files can be copied directly to other MySQL servers, it waits for any DML query to execute completely using locks for

the table and then only next query can be processed, supports indexing and full text searches along with speed read operations. Due to some advantages rather more disadvantages InnoDB had became indeed mostly used MySQL engine. Major features of InnoDB includes supporting ACID (Atomicity, Consistency, Isolation and Durability) principles which means each transaction is treated separately due to row level locking, along side supports transactions including rollbacks, automatic data recovery in case of crashes, supports foreign keys, has an advanced set of datatypes and faster for DML queries while for DQL the speed remains mostly same to MyISAM. Due to all these features and added advantages InnoDB is mostly preferred for selecting a MySQL database engine though only occasionally this vary under requirements. Though as a good practice while creating a database I would suggest using InnoDB generally. By keeping this mind lets move on to the next section which is user management.

User management is an important part of MySQL and specific users can be provided access to specific database, operations or hosts which restricts their usage to not needed data and schemas. Lets now dive deeper into how user management is done in MySQL. Firstly when we install MySQL a default user is installed named as root for which we can set a password or not during installation. This user has every access for the server. So for initial operations into the server we need to login using this user.

After logging in we can create users and provide access rights to them specific to databases or hosts. Lets look into this now. Very firstly lets give a look into the query that can be used to create a database user in MySQL.

CREATE USER <NAME>@<HOST> IDENTIFIED BY <PASSWORD>

This is followed by granting privileges to the user created. This can be done by granting all privileges or specific privileges to the user. Lets see the command to grant all privileges to the MySQL user created

GRANT ALL PRIVILEGES ON <DATABASE.TABLES> TO <NAME>@<HOST>;

This user will get all privileges to all the tables to the database provided. In case of specific tables that can be provided or for all tables wildcard (*) can be used. We can also provide specific privileges to the user for the database that can be SELECT, INSERT etc. For providing specific privileges to the user for the database we can provide this as below.

GRANT SELECT ON <DATABASE.NAME> TO <NAME>@<HOST>;

We can allow multiple operations to the user by using comma separated in the query (SELECT, INSERT…). The various permissions available to be provided are SELECT,

INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, INDEX, GRANT OPTION, FILE, RELOAD, SUPER and SHUTDOWN. These are all the permissions that can be provided on the user. For providing rights for all available databases to the user we can use wildcard (*) in place of the database. Except these there are many privileges which are MySQL specific which are CREATE USER, CREATE ROLE, DROP ROLE, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, EXECUTE, EVENT, TRIGGER, REPLICATION SAVE, REPLICATION CLIENT, BACKUP_ADMIN, AUDIT_ADMIN, CONNECTION_ADMIN etc. To look all these queries to set these privileges will make the chapter lengthy. So we shall look into the most important ones here and guide you through the process of the same.

Similarly in the query we can specify hosts or tables for providing the access for them only. To note in a query we can provide only one table name or the wildcard for all the tables. To provide access to multiple table we can write multiple queries. So once these queries are executed the user would be created and will have specified permissions for database or MySQL server as a whole. Following this we need to use the FLUSH command to complete the query and release the MySQL user for access for connections through the server. Below is the syntax of the command.

FLUSH PRIVELEGES;

Once the user creation is completed we can check the various permissions available for the user with the SHOW GRANTS command. Below is how this command is being used.

SHOW GRANTS FOR <NAME>@<HOST>;

This command will show the user access this user has along with the hosts and databases it has the accesses to it. So this is the whole process how a user can be created and granted access to specific databases or hosts. You can also note that there are specific UI which are already available if we look to create them from any MySQL client IDE like MySQL Workbench, PHP My Admin etc. For PHP My Admin this can be created by navigating to the Users section and following the steps through the UI. So this is a brief on how users can be created for the MySQL for specific databases, hosts or even server level accesses. Now lets move on and look into supported data types that are available in MySQL 8.0.

To learn more into the available data types lets first dive into and look the available data types that are in MySQL 8.0. Major of all the datatypes are listed as follows and we shall afterwards look into what each of them actually means. So here is the list of major available data types in MySQL 8.0.

- INT
- BIGINT
- BIT
- DECIMAL
- FLOAT
- DOUBLE
- CHAR
- VARCHAR
- BLOB
- TEXT
- ENUM
- DATE
- TIME
- DATETIME
- TIMESTAMP
- BOOLEAN
- JSON
- GEOMETRY

So these are most of the major data types that are used in MySQL 8.0. Let's now dive into and look at what they means and where they are actually used. And so lets begin with the first data type in the list INT.

INT is used to store data that are of type integers. The range of INT typically is from -2147483648 to 2147483648. The syntax used to define a column for data type INT is as below.

<column> INT <auto_increment> <key> <unsigned>

We can use this syntax to declare a column of int data type inside the create table SQL query. There are few optional parameters that we can also pass besides the data type which can be signed/unsigned, keys (unique/primary/…), auto increment etc as per requirements. Now this is how INT is declared. Lets move on to the next datatype BIGINT.

BIGINT is similar to INT that it can store integer data but it has a greater range. The range of BIGINT varies from -9223372036854775808 to 9223372036854775808. This is why it is more suitable for large scale real world applications. The syntax of BIGINT is also similar to INT as is mentioned below.

<column> BIGINT <auto_increment> <key> <unsigned>

Now this concludes that how we can declare BIGINT. Lets move on to the next data type which is BIT. A bit is very important in computer programming and is the basic operator in low level operations. In MySQL bit has been introduced as a datatype which generally takes two values 1(On) and 0(Off). It is very important in maintaining logic in MySQL and other programming languages. It is declared as below in a MySQL table.

<column> BIT <null>

Here the above optional parameters are not required as a bit is not suitable for auto increment or as a key rather it can be stated as nullable or not. Now lets move on to the next data type which is DECIMAL. DECIMAL is used to mention a number up to some decimal points. This is generally represented as DECIMAL(M, D) where M stands for the number before the decimal point which can be ranging from 1 to 65 whereas the D stands for the number of digits after the decimal point which can range from 1 to 30. If they are omitted the default values for M is 10 and D is 0. It can take optional parameters like NULL or ZEROFILL or some other parameter. ZEROFILL is for padding the numbers after decimal with zeros for maintaining the number length. Below is the syntax of declaring DECIMAL numbers.

<column> DECIMAL(M, D) <null>

Now lets move on to the next data type FLOAT. FLOAT is named for floating point numbers and are used to store decimal numbers with a optional parameter for precision. Following is the syntax for FLOAT.

<column> FLOAT([M]) <null>

Here M is the optional parameter for FLOAT which dictates how many digits precision after the decimal point is required. The data type can be used without the parameter as well. Except we can specify other optional parameters like null etc. Now lets look into DOUBLE which is similar to FLOAT with capability of storing more precise values.

To declare DOUBLE we can use the below syntax. DOUBLE is same as FLOAT except it is more precise. This means it can hold numbers with more digits after decimal place.

<column> DOUBLE <null>

Now lets move and check next important data type CHAR. CHAR is a data type that is used to store character values of fixed length. We declare the parameter M while declaring CHAR data type which defines how many character length of data it can store.

The maximum possible value of M is 255 and by default it is 1. The point to note that for CHAR the length of the string is fixed and cannot be changed. Following is the syntax how a CHAR data type can be used.

<column> CHAR([M]) <null>

The next data type which is similar to CHAR is VARCHAR. VARCHAR differs from CHAR in the sense that it can hold variable length values and also it has larger character length which it can store. We can mention a maximum length for VARCHAR which it can hold and it can hold any number of characters between 0 and that value. Also VARCHAR has a larger length that it can hold than CHAR. We can declare VARCHAR as follows.

<column> VARCHAR([M]) <null>

We can provide other optional parameters to VARCHAR like NULL or DEFAULT values. Here M denotes the maximum length of character that it can hold. So that's about VARCHAR data type. Now let's move on and look into next in the list the BLOB data type.

BLOB is a data type that is used to store binary data. This is specifically used to store data like images, files as it is in the database. Though it consumes quite data it is more secure and easy to process comparative to storing paths. So we can story binary data in records of type BLOB. Lets see the syntax used for declaring BLOB.

<column> BLOB <null>

So this is how BLOB is declared and as usual we can specify optional parameters while declaring the column. Now lets check out what can be done using TEXT.

TEXT is a very important data type and can be used to store variable long length character data. TEXT has a limit of 65535 characters of maximum length. TEXT has variations that can store smaller or larger amount of data which are TINYTEXT, MEDIUMTEXT or LONGTEXT. LONGTEXT has very high storage capacity. TEXT generally contains variable amount of string within it. Below is the syntax used to store TEXT.

<column> TEXT <null> <default>

We can provide other optional parameters like NULL or DEFAULT along with the data type while declaring the column. Now lets look into the next data type which is ENUM.

ENUM is used where we have a set of data that can only be stored in the column. We need to pass the set of values while declaring the ENUM data type. The maximum number of values possible in the set is 65535. The values can be string, character, integer etc. Below is how an ENUM is declared.

<column> ENUM([A,B,C…]) <null>

Here the values are defined using comma separated elements which can be string etc. The value when entered into the column must be within these set of values. Here also we can declare a ENUM as nullable column. ENUM is very useful when a column should contain data from a set of values. Hence now we shall move on and look into the various date and time data types. Among them we shall start with the data type DATE itself.

DATE is a MySQL data type that stores the date in the table. The date format used is YYYY-MM-DD. This is the default format and we can declare DATE data type for a column as any other data types. Below is the syntax that is used to declare a column as DATE data type. Here also we can specify the column as nullable or a default value for it.

<column> DATE <null> <default>

There are many inbuilt functions in MySQL which can assign the date value to the specific column directly. We shall look into the inbuilt functions later in this chapter. But to keep in mind these functions can take the value from where the MySQL server is running as date and time values specifically depends on the location and system on which MySQL is running. Now lets move and look into TIME.

TIME is a data type which is sued to store time in MYSQL column. The default format of TIME is HH:MM:SS which is in an 24 hour format. Below is the syntax that is used to declare a column as TIME data type.

<column> TIME <null> <default>

We can use nullable and default values for data type TIME as well. Now lets move on to next data type which is most widely used for date and time functions and that is the DATETIME data type.

DATETIME data type is used to store date and time together in the database. This is the most widely used data type regarding date and time because it is important for most real world applications. The format for date and time is YYYY-MM-DD HH:MM:SS where time is in 24 hour format. Below is how we can declare a DATETIME data type.

<column> DATETIME <null> <default>

We can pass some optional parameters to the declaration while this is most commonly used data type to store date and time values. Now lets move on to next data type which is a bit different but more accurate in storing time which is TIMESTAMP.

Generally TIMESTAMP refers to the difference in seconds for the mentioned date and time from 1st January 1970. In modern MySQL versions TIMESTAMP returns date and time value only while to get difference in seconds we need to use UNIX_TIMESTAMP() function. The major drawbacks with TIMESTAMP data type is it can store values starting 1970 till 2038. Any illegal date and time value or outside the range will save the TIMESTAMP as 0000-00-00 00:00:00 if MySQL zero mode had been enabled for the server. Now lets look into how it is declared.

<column> TIMESTAMP <null> <default>

For all the date and time data types one of the major default values used are CURRENT_TIMESTAMP while they are used to store the record insert and update time. We shall look into these cases later. But here also we can use the optional parameters. So these are all the major date and time data types used in MySQL. Now lets move on and look into the BOOLEAN data type.

BOOLEAN is a data type which consists of two values true and false. Practically any value that is either numerical 0 or FALSE is considered as false and all other values are considered true which includes 1 or TRUE. It can also be said as a synonym of TINYINT(1). Lets check how it is declared.

<column> BOOLEAN <null> <default>

We can pass other parameters like NULL etc also. BOOLEAN data types are mostly used to save specific cases which results in either true or false values. Next lets look into another very important data type in MySQL introduced recently JSON.

JSON is one of very important ways to transfer data between endpoints as data in these cases can be stored together without breaking their structure. So storing JSON data in column as it is provides much more flexibility to the developers. There are important functions like JSON_EXTRACT() which can be used to process JSON data in MySQL queries itself. Let’s look how a JSON data type is declared.

<column> JSON <null>

We can also directly extract various values from the structured data using $.<name>. We shall look into it later. Now lets move on and look into the next data type which is GEOMETRY.

GEOMETRY is a spatial data type introduced in recent versions of MySQL. This can be used to store structure of objects. Spatial indexes are important for smooth processing of these columns. Also data must be inserted using special functions like ST_GeomFromText(), POINT() etc over text strings to safely save the data. Lets look into its declaration below.

<column> GEOMETRY <null> <spatial_indexes>

So like this we can save records into GEOMETRY data type columns. Hence this concludes the section of data types in MySQL as we have covered most of the major data types. We shall look later on these and other data types when required. Now lets move on to the next section which is keys and indexes.

Indexes and keys are very important aspect in MySQL in respect of performance and data management. Indexed columns makes search queries much faster specially for engines which supports transactions. Again keys make data linking and redundancy more accurate. Lets dive deeper into it and start with indexes.

So how indexes works? It creates references to rows in a table and according to the query it makes the retrieval faster based on the reference. Basically indexes are of these following types.

- Primary Indexes
- Unique Indexes
- Full Text Indexes
- Composite Indexes
- Spatial Indexes

We shall look into each of them in detail. Primary indexes forms the primary key of a table and they are used to uniquely identify each row in a table. They should be unique and cannot be null. They are very important for identifying rows and further speed up query executions. Most common type of primary indexes are columns with Auto Increment values which are easy and convenient for larger tables. Below is how an index can be assigned to a table in SQL language.

CREATE INDEX <index_name> ON <table_name> (<column_name>)

Next lets look on to Unique indexes. Unique indexes can be applied to columns or a combination of columns which are unique. This can be used to speed up searching across those columns. In a table there can be multiple columns that are unique but for primary this is not the case and only one column can be used for that. Also Unique

indexed columns allows NULL values. Now lets look how a unique index can be defined.

CREATE UNIQUE INDEX <index_name> ON <table_name> (<column_name>)

To note that any of the indexes can be defined while creating a table or by altering it using CREATE TABLE or ALTER TABLE queries respectively. Also these are the most used ways to define indexes for a table while it is created. Lets move on and look into FULL TEXT indexes now.

A FULL TEXT index is a type of index which is used to make faster searching on text based columns like CHAR, VARCHAR and TEXT. It makes the searching faster based on specific search terms. Below is the syntax of defining a FULL TEXT index.

CREATE FULLTEXT INDEX <index_name> ON <table_name> (<column_name>)

They can also be created on multiple columns and also while creating and altering the table schema as discussed earlier. Now lets look into COMPOSITE index. A COMPOSITE index is an index that is a combination of two or more columns. This can be unique or any other indexes. This indexing can be used for speed, data identification etc. Below is the syntax of COMPOSITE index.

CREATE INDEX <index_name> ON <table_name> (<column1>,<column2>…)

This can be declared on UNIQUE index, PRIMARY key or even while creating or altering tables. So these is how indexes can be created. Now lets move on to next index which is a SPATIAL index that is used with SPATIAL data types.

SPATIAL indexes are used to fasten searches over columns with SPATIAL data types. As SPATIAL data types are special data types with different formats of data being stored hence SPATIAL indexes can work over these columns only. Lets look how a SPATIAL index is declared.

CREATE SPATIAL INDEX <index_name> ON <table_name> (<column_name>)

This can be declared while creating or altering tables as well but one thing to note is that they can be applied on columns which must be NOT NULL. Now as we finished looking into indexes lets now look into what are keys and their relation with indexes.

So keys are something in MySQL which are used to set up some constraints or used for identifying columns separately in MySQL. So lets now list the various keys available in MySQL. Below is the list of various keys that are available in MySQL.

- Primary Key
- Unique Key
- Foreign Key
- Composite Key

Lets first discuss the most important among all PRIMARY key. A PRIMARY key is something that in fact is identity of the table and is used to uniquely identify a record. Generally PRIMARY keys are auto increment columns that are created specially for this purpose. Though any other column can be used as a PRIMARY key but on basis that this column should be unique or else while entering a record if a duplicate value is entered for the cell then the row will not be rejected and the SQL query returns a error. Lets now look how a primary key can be declared. Generally it is declared while creating a table within the create table query.

<column_name> <data_type> PRIMARY KEY,

This should be within a create table command though there are other ways PRIMARY keys can be declared as well while altering table schema or for composite primary keys. We shall look at them when required. Now lets move on and lets look on to UNIQUE key.

Though unlikely PRIMARY key, UNIQUE keys can be multiple in a table and are used to maintain uniqueness in all the records in a column. UNIQUE keys helps besides identifying rows uniquely also in indexing and as a result faster searching. UNIQUE keys though also allows NULL values. Lets see how a UNIQUE key is declared.

<column_name> <data_type> UNIQUE <null>,

UNIQUE keys can be declared while creating or altering tables or while using ADD CONSTRAINT as well. Again UNIQUE keys can be composite as well. So that's all about UNIQUE keys. Now lets take a look into FOREIGN keys another important feature in MySQL.

FOREIGN keys are keys that generally create a relation of a column in one table to column in another table. It actually enforces "referential integrity". Few terms in this respect are parent table, child table and referential integrity. What actually happens is that some primary or unique key in parent table is linked with some column or columns in child table on the basis of referential integrity. Now while we update, delete or perform some other action on the parent or master table the child table also would get affect subsequently on the basis of the referential integrity. Like for an example if we delete the record from parent table then in child table the row will get deleted having column with the same value as mapped with the column with parent table. This is how

foreign key works and is helpful in maintaining data redundancy. Lets look how a foreign key is declared.

CONSTRAINT <foreign_key_name> FOREIGN KEY (<parent_column_name>) REFERENCES <child_table_name>(<child_column_name>)

This is used while creating a parent table in MySQL. FOREIGN keys can be created while altering tables or separately as a constraint as well. Few of the operations that can be performed using FOREIGN keys are CASCADE (delete / update the record as per operation on parent), SET NULL (sets the child record NULL), NO ACTION (prevents delete / update if record exists in child table). These actions can be performed while ON UPDATE or ON DELETE operations. We shall look more into it later. Lets move on and look into the final key which is a COMPOSITE key.

COMPOSITE keys are keys that are formed by combination of two or more columns. COMPOSITE keys can be PRIMARY or UNIQUE. Lets check the syntax how a COMPOSITE key is declared. Below is actually an example of COMPOSITE primary key.

PRIMARY KEY (<column1>, <column2>),

This is how a PRIMARY COMPOSITE key is declared while creating a table where column1 and column2 would be declared generally as already discussed. Similarly COMPOSITE keys can declared while altering table or using ADD CONSTRAINT. Also to be noted that all the discussed keys that be dropped using the below command.

DROP <key_type> <key_name>,

These keys are dropped by altering the table. All keys can be provided or by default assigned specific names and that names need to be passed for dropping a key. No two keys in a table can have same name. So that's all on various keys available in MySQL. Lets move on the next section now.

Now we shall discuss about various MySQL queries generally known as Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Language (DQL) queries. We shall look into each of them one by one and lets start with DDL queries. DDL or Data Definition Language queries are queries that are used to define the structure that will hold the data like create table queries.

Lets dive deep into DDL queries. In addition to defining the table structures they can alter, delete or modify the schemas. These are the major tasks that are performed by DDL queries. Though there a lot bunch of DDL queries here we shall look into the

major ones. Below is the list of the major DDL queries which constitutes an important part in MySQL.

- CREATE
- ALTER
- DROP
- TRUNCATE
- RENAME

So lets start with the CREATE DATABASE / TABLE command. The very first step in MySQL is to create a database. So we shall firstly look into creating a database followed by creating a table in the database. To create a database we need CREATE DATABASE query. To create a database we simply can use this below command in any of the MySQL prompt.

CREATE DATABASE <database_name>;

This will create a database with the specified name. To view all databases in the MySQL instance we can use the command SHOW DATABASES. This will list all the available databases in the instance. We shall look later the command and how to actually delete or drop a database. Lets move on and now look into next CREATE command which is the CREATE TABLE command.

CREATE TABLE command is used to create a table in the specified database. So after creating a database we need to select the database to perform further operations into the database or rather pass the database name along with the table name to specifically identify it. In a database no two tables can be assigned the same name but in different databases it can be. So to select a database we can use the below command.

USE <database_name>

This will select the specific database and we can perform further operations for or into that specific database only. So here lets look and start with the CREATE TABLE command. This command is specifically for creating table schemas in the MySQL database to store data. We need to decide what and which type of data should we store in the particular table and create the table using the query accordingly. Lets look into a simple CREATE TABLE command.

CREATE TABLE <table_name> {

<column_name_1> <data_type> <constraints…>

<column_name_2> <data_type> <constraints…>

…

}

So this is a basic structure of using the CREATE TABLE command. We have checked earlier on how to specify data types and declare columns. We need to enclose the same inside the command to create the table schema. We can specify various keywords while creating a table to check if a table with same name already do not exists or to clone a table etc. We can use IF NOT EXISTS and LIKE <parent_table_name> sort of keywords for the same respectively. Now lets move on and check how to modify a table schema once its created using ALTER TABLE command.

ALTER TABLE command is used to modify table schema. In case the table has data the options needs to be consistent with the data or the command would throw an error. While adding or dropping a column simple ADD or DROP COLUMN can be used but to be noted if the table has data a default or null value is a good practice in case of adding column. Lets look into these two commands followed by we shall see on how to modify a column in MYSQL.

ALTER TABLE <table_name> ADD <column_name> <data_type> [<options>]

This is how a new column can be added to a table. A default or null value is preferred in case there is already data in the table for existing rows. Now lets check on the DROP command.

ALTER TABLE <table_name> DROP COLUMN <column_name>

This is how to remove a already existing column from a table. Its simple and easy to use without much variations. Now lets look into how a column can be modified in a table. In case there is data and we need to modify data type of the column it needs to comply with the data.

ALTER TABLE <table_name> MODIFY COLUMN <column_name> <data_type> [<options>]

This is the command on how to modify a column and we can provide any new index or keys or change the name or data type of the column. ALTER TABLE though is important but also tedious and hence as a best practice is advised to create a table initially as per requirements after thorough planning and later instead of modifying the table create new tables linked by any of the keys. Lets now move on and look into the steps to delete a table. The command that is used to drop a table is the DROP TABLE command. We shall look into it now.

DROP TABLE is simple to use alongside it also provides some options to use like check if table exists using IF EXISTS and also dropping multiple tables together. Below is the syntax for dropping tables.

DROP TABLE [<IF_EXISTS>] <table_name_1>[,<…more_tables>]

This is simple to use and understand and is used to totally delete the table schema along with the data. Now lets look into how we can clean up all the data in the table along with setting auto increments to default or in other words reset the table. The command used to perform that is the TRUNCATE TABLE command. Lets look into it below.

TRUNCATE TABLE command as discussed is used to delete all the data along with resetting the indices of the table. Below is the syntax of TRUNCATE TABLE command. One point to note is that this command does not activate the triggers that are set on ON DELETE for the table.

TRUNCATE TABLE <table_name>

One alternative to TRUNCATE TABLE is to use DELETE FROM without any options. Though that is in Data Manipulation Language queries and that behaves somewhat differently from truncate command. Also once TRUNCATE TABLE is used that cannot be rolled back for even engines supporting transactions. Hence that's all for TRUNCATE TABLE. Now lets look how we can rename a table using RENAME TABLE command.

RENAME TABLE command is used for renaming tables. It can be used to rename one or multiple tables on the go. Below is the syntax for renaming tables.

RENAME TABLE <old_table_name> TO <new_table_name>, …

Same syntax can be used to rename multiple tables using comma separated queries. Also we can transfer a table from one database to another database using the same command as below though only on same server.

RENAME TABLE <database_one>.<table_one> TO <database_two>.<table_two>;

These are some of the usages of rename table. So these are the important queries in our list of Data Definition Language queries. There are some other things though as we are aiming to provide basic or in fact go over important things in any of the languages as we focus on guiding on creating complete projects and the content provided is up to for the purpose. Though if required you can go through other uncovered topics from the internet or through complete reference books. So this concludes the Data Definition

Language part now we shall move on and look into various of the Data Manipulation Languages shortly referred as DML queries.

Data Manipulation Language queries are used to modify records in a table basically by performing insert, update or delete. There a lot of other methodologies which we can use along side MySQL queries are classified broadly as control-flow, clause etc. We shall look briefly into them too later in this chapter. Now lets move on and look into the basic DML queries listed below.

- INSERT
- UPDATE
- DELETE

Lets look into INSERT queries in the beginning which are basically used to enter new records in a table. INSERT queries are mainly used in two ways and also for entering single or multiple records at once as per necessity. Lets look first into the most basic type of usage of INSERT query.

INSERT INTO <table_name> (<col1>, …) VALUES (<val1>, …)

This above syntax is used for inserting a single record into the table with the specified table name and columns. We need to provide the list of columns along with similar number of values. To be noted if a column does not have default value then it must be provided in the INSERT statement otherwise the query will return an error. Also we can leave if there is an AUTO INCREMENT column in the table. While lets look how to insert multiple records together.

INSERT INTO <table_name> (<col1>, …) VALUES (<val1>, …), (<val1>, …), …

This above syntax is used for inserting multiple records together to the table. There is a separate syntax using the SET keyword also to insert records. Lets look into the syntax below.

INSERT INTO <table_name> SET <col1> = <val1>, … ;

In this syntax the column value pairs are passed one after the other. Another way is used for inserting records into a table from another table. This is used performed using the SELECT statement.

INSERT INTO <table_name> (<col1>, …) SELECT <col1>, … FROM <from_table_name> WHERE <condition>;

This is a short way by which records can be inserted from some table to some other table. Except these we can use some special options like INSERT IGNORE or INSERT … ON DUPLICATE KEY UPDATE etc which performs functions of skipping if the record entry returns an error, update the record in case of duplicate key respectively. There are many other variations in the INSERT method as well which we shall look into as and when required. Also you can go through them over the web but mostly the described methods can do our task most of the time. So this is all about the INSERT statement. Now lets move on and look into the UPDATE statement.

UPDATE statements are used to update data that is already present in the table. This must be used with a condition else otherwise if required to update all the available records in the table. Lets look below into the syntax of UPDATE query.

UPDATE <table_name> SET <column> = <value>, … WHERE <condition>

Here we can update multiple columns simultaneously on the basis of the condition used. There if we want to update only one record we can perform that based on the condition of primary key column. Also we can use methods like update a column value based on its previous value etc according to the necessity. While used with PHP this is one of the highly used DML queries to update specific already saved records. So now lets look into the next DML query which is DELETE. To be noted one of the best practices is that to use soft deletes which is updating the record and setting a column valid (or any other suitable name) to false. This keeps the record for any future requirements while still able retrieve only valid records from the table. On the other case DELETE query deletes the complete record with no traces for any future requirements.

So lets look into the DELETE query and how it is used. DELETE query is suggested to be used with a condition otherwise if not all the table data can get deleted. But in case for deleting all table data together as we discussed TRUNCATE is a better option as it sets the indexers to its default values too and resets the table. But lets look into how the DELETE query is written now.

DELETE FROM <table_name> WHERE <condition>

By using the above query format we can delete multiple data from the table according to the condition specified. Though there are certain modification available like using ORDER BY and LIMIT along with the conditions. Using them we can delete the first specific number of records mentioned by limit. We can use COMMIT and ROLLBACK before COMMIT in case of DELETE operation which is though not possible for TRUNCATE queries.

So with this we conclude the Data Manipulation Language (DML) queries list. Now lets move on and look into the other type of query operations which are Data Query Language (DQL) operations. DQL queries are mostly used to retrieve data from table on the basis of specific conditions mentioned and are most common in MySQL and PHP applications. Now lets look into them in details.

Below is the list of DQL queries. With this list we shall look into basic options that are used in the conditions for retrieving the data. Lets look into the basic DQL queries list.

- SELECT

SELECT is the only and mostly used DQL query which is used with variety of conditions which we shall look into now. SELECT is used to fetch all records or based out of conditions from tables. Lets look into the basic syntax then we shall look into what various conditions it can be used with.

SELECT <columns…> FROM <table_name> WHERE <condition>;

To note that SELECT can be used with one table or multiple tables using JOINS or also in a SELECT query we can fetch data from other tables too using sub queries. Lets look into examples of JOINS and sub queries.

For JOIN we can use as below.

SELECT <t1>.<columns…>, <t2>.<columns…> FROM <table1> <t1> JOIN <table2> <t2> ON <t1>.<col1> = <t2>.<clo1> WHERE <condition>;

In this query two tables are joined using the ON condition where the values of col1 s matched in tables table1 and table2. All those records are fetched which are matched as per the ON condition. We can use various types of JOIN like INNER JOIN, LEFT JOIN, OUTER JOIN etc. which are used for fetching records even if matching condition not exists in other table or only if only exists in both tables. The various types of JOIN can be checked in details from books that are specific for MySQL. We shall move on and look into sub queries and various conditions available in MySQL now.

For sub queries we can use as below.

SELECT (SELECT <col1> FROM <table2> WHERE <cid> = <table1>.<col1>) As <alias1>, <col1>, … FROM <table1> WHERE <condition>;

Here the sub query is selected as an alias and returned with the records from the primary SELECT query. Using sub queries is suitable in some cases whereas using joins

in some cases. Now lets move on and look into various other entities that we can use with SELECT queries conditions like ORDER BY, GROUP BY, HAVING, LIMIT, IN, LIKE, AND / OR / NOT, BETWEEN, IS NULL, UNION etc. We can use If, CASE etc for conditional fetching as well in MySQL. Besides these we can use several functions like SUM, COUNT, MAX, UNIQUE etc for fetching records as per requirements. We shall look into some of these now and for details any MYSQL specific book can be referred. Lets look into them now.

ORDER BY is used to sort records based on conditions and can be used with the column name followed by the parameter ASC or DESC as required. The syntax of using ORDER by is as ORDER BY <column_name> <ASC>/<DESC> used after the WHERE condition. We can limit also the number of records to be fetch using the LIMIT keyword. It is used as LIMIT <start_index>, <no_of_records> which is often use at end of any SELECT query.

GROUP BY is used for grouping records into sets identified by a specific column value. It is used as GROUP BY <column_name> where all records are returned in sets of specific values of the column and often the last record is fetched for column in SELECT clause if we do not use any aggregate function like AVG, SUM, COUNT etc in the SELECT statement for fetching the records. After using GROUP BY we use HAVING to filter out records based on condition instead of WHERE as for general SELECT queries. The syntax of GROUP BY queries are as below.

SELECT <columns…> FROM <table_name> WHERE <condition> GROUP BY <col1> HAVING <group_by_condition> …;

In WHERE conditions we can match records for specific columns using IN (for matching with records from an array), LIKE (for matching part of a string) etc. We use operators AND, OR, NOT for combining various conditions together in the WHERE clause. Mostly for date time columns we use BETWEEN to check if the date and time column value lies between two specific date-time values. We use operators like =, >, <, and value NULL to check the records as per conditions as well. Some of the important functions we use are UNIQUE (only unique records for the column), SUM (sum of the records of the column), COUNT (no of records for the column), MAX (maximum value of the column) and so on. So these are some of the entities that are used with SELECT queries.

There are some more advanced entities that are often used in MySQL are if clauses, switch clauses and UNIONS which are used in WHERE condition. If clauses are used with following syntax – if (<condition>, <value_if_true>, <value_if_false>). CASE clauses are used as CASE WHEN <cond1> THEN <val1> WHEN <cond2> THEN <val2> ELSE <val3>. While UNIONS are used to combine records fetched from two

queries and return the result into one set. So these are some of the major entities that are often used and shall be required while developing apps with PHP as well or even if we use MySQL stand alone too. For looking into complete details of features that SELECT queries offer any book with complete reference to MySQL can be referred.

As our goal in this book is to make the reader able to develop complete business applications using PHP, Smarty and MySQL we shall move on and look into some of the other features that MySQL offers before moving on to the next chapter. So as our next section is transactions in MySQL we shall move on and look into that now.

Transactions in MySQL are a way to ensure ACID properties to ensure data integrity. It enhances Atomicity, Consistency, Isolation and Durability in the records. To ensure this we use a set of statements to maintain the integrity and each such cycle is called transactions. Transaction can include operations on one record or multiple records. One thing to note is that still a transaction is not completed we can ROLLBACK the operations but once its completed we can not. We can also set AUTOCOMMIT value to 1 to enable each operation to be treated as a complete transaction. Lets look into transactions in details but before that list down major statements used in transactions after which we shall start with looking into its flow.

- START TRANSACTION
- COMMIT
- ROLLBACK
- SAVEPOINT
- SET AUTOCOMMIT

Now lets start discussing transactions with the flow of how the statements are used. This will be followed by we shall look into why and how we use them. The first statement we use while using transactions is START TRANSACTION. This is used before any of the queries in transaction and any query written after this and COMMIT is not made resembles to the transaction and can be ROLLBACK till COMMIT is made. ROLLBACK is used to revert the changes till the last COMMIT. To note is that COMMIT make a final changes according to the queries after which ROLLBACK can not be used on the queries. While SAVEPOINT is used as a temporary save point after using the queries can be reverted to that point using the ROLLBACK command. These are some of the statements used in transactions in MySQL. Whereas we can use SET AUTOMMIT = 1 to auto commit each query used in the MySQL server automatically. So these are some of the ways how transactions can be used in MySQL.

Now as we know MySQL has a way to have some special types by which queries can be executed. Lets name them first after which we shall look into what they are and how they can be used. So lets look into the various types of these special ways listed below.

- Functions
- Triggers
- Procedures
- Events

So lets start with Functions. Functions are typically stored methods in MySQL or database which can be used with SELECT or any other Query in MySQL. Lets look into it in detail. There are many predefined functions that MySQL already has whereas user defined functions are also supported. Some of the predefined functions are like SUM, AVG, MAX, CONCAT, REPLACE, STRING, ABS, CEIL, NOW, YEAR, LENGTH etc. All these functions work differently and are useful in variety of queries. Now we shall look into a variety of user defined functions and how we can define and use them. Firstly lets look into the way we can define user defined functions in MySQL and its syntax.

```
DELIMITER <temp_delimiter>

CREATE FUNCTION <function_name> (<parameter_one> <datatype_one>, …)

RETURNS <datatype>

DETERMINISTIC

BEGIN

    <function_body>

END <temp_delimiter>

DELIMITER ;
```

So this is how a function is defined in MySQL. Lets go through and look into what each line and statement means and is used for. DELIMITER is used to temporary change the statement delimiter to some other value so that semi colon (;) can be used within the function body. Next we define the function name and the parameter list it can accept along with the data types specific to each. This is followed by specifying the data type of the return value it will. Then we specify the function would be DETERMINISTIC or NONDETERMINISTIC that function will return same output for same input or it can vary or be random in the sense. This is followed by the BEGIN keyword after which the function body will be written. We end the function body using END and defined delimiter. In the end we change the DELIMITER back to semi colon (;), While in the function body we use different keywords like DECLARE (to declare a variable), RETURN (to return a value) etc as per need. So this is how a function is

generally declared and saved in a MySQL database. This is followed by the function can be called from various queries in the database. These are called UDF functions also in short resembling user defined functions. Now lets look how they can be called from a query.

These functions can be called like any other built in functions from various queries. Lets look into the syntax of the same below.

SELECT <function_name> (<parameter_one>, …) AS <als> FROM <table_name>;

So this is how a user defined function can be called from MySQL. We can pass a alias to the output of the function to be displayed in the result of the query. So this is the basic of how functions are used in MySQL. Also to note to delete a function we can use the query DROP FUNCTION [IF EXISTS] <function_name>. Lets look now into what are triggers and how they are used in MySQL.

To describe triggers are specifically stored procedures that are executed when a specific event on table is executed like INSERT, UPDATE or DELETE. These are necessary at many times when we need to perform something in background to update records and not even in the business logic. There are basically six types of triggers which are executed BEFORE or AFTER these three operations INSERT, UPDATE and DELETE. Lets look into one with its syntax. The rest can also be defined similarly. So below is the syntax of a trigger.

```
DELIMITER $$

CREATE TRIGGER <name> <BEFORE/AFTER> <INSERT/UPDATE/DELETE>

ON <table_name> FOR EACH ROW

BEGIN

    <trigger_body>

END$$

DELIMITER ;
```

Here DELIMITER is similar as in case of functions. To note that in the CREATE TRIGGER line we specify the trigger name and when the trigger to be executed. Points to note are that we can not use DDL queries in trigger, can perform operation on other tables, can not truncate table and we can use NEW.<col_name> or OLD.<col_name> for referencing data for the table on which basis the trigger is executed. By keeping these

points in view we can write triggers on various requirements following the above syntax. So now lets move on procedures and lets look what are they and how they are executed. Also along with that lets look what is its syntax and when it is required and how to call it. For triggers they are executed when the condition met as specified while declaring the trigger. Also note that to delete triggers we can use the query as DROP TRIGGER [IF EXISTS] <trigger_name>.

Lets look into procedures now. Procedures in MySQL are something that are like functions but do not return a value. In procedures we need to pass variables called as IN, OUT, INOUT. Rest mostly its similar like functions except how to call the. We shall look at that later while lets now look how to declare a procedure. Below is the syntax of a procedure.

```
DELIMITER $$

CREATE PROCEDURE <procedure_name> (

    IN <param_one> <datatype_one>,

    OUT <param_two> <datatype_two>

)

BEGIN

    <procedure_body>

END

DELIMITER ;
```

Here to note that IN, OUT or INOUT are to define if the variable is a input variable, output variable or can be both. Rest they are executed using the CALL statement. Lets look into that below.

```
CALL <procedure_name>(<param_one_val>, @<out_param>);
```

Using the above statement the output variable, <param_two> here will be assigned to the @<out_param> in the MySQL for the database. For checking the value we have to use the SELECT statement as described below.

```
SELECT @<out_param>;
```

So this is how procedures are used. To note we need to pass the output to the parameter mentioned as OUT in the procedure to retrieve the value from it and only OUT procedures can send back the value. IN parameters are used to pass values to the procedure while INOUT parameters are also there which can be used for both purpose. So this is how procedures can be used. Note to delete a procedure we can do in similar fashion like DROP PROCEDURE [IF EXISTS] <procedure_name>. Lets move on and now look into what events are and how to define and use them.

Events are something that are automated tasks that runs on schedule in MySQL. To enable events the event_scheduler must be set to ON if not. This can be done by the statement SET GLOBAL event_scheduler = ON; There are two types of events one time and recurring usually differentiated by the EVERY keyword. Lets look below how an event is declared followed by which we shall look various topics on it.

CREATE EVENT <event_name>

ON SCHEDULE AT <date_time> / ON SCHEDULE EVERY <d> DAY STARTS <date_time>

DO

<event_body>

Here we can look into both type of events one using AT is a one time event while using EVERY is a recurring event. We can select as appropriate. Rest of the body we need to specify event name and the event body which is similar to as in function / trigger / procedure. So this is how events are used. We can set the event scheduler ON permanently using my.cnf for MySQL. While to delete an event we can use DROP EVENT <event_name>. Also we can alter events using ALTER statement. Events are mostly used for backups or even generating reports. So this is how we can use functions, triggers, procedures and events in MySQL. Now lets move on the next section in this chapter which is how we can use MySQL in PHP and followed by it we shall look into some of the best practices.

While using MySQL in PHP most of the time DDL queries are least required while DQL and DML queries are quite often used. In PHP earlier there are two inbuilt libraries that are available to use MySQL. But in PHP8 and later only one library is available that is the mysqli which provides two ways – procedural and object oriented approaches to use mysqli for connecting and inserting, modifying data in databases. It is better to create database and table schemas directly from MySQL but sometimes we require to use the DDL queries also within PHP. But these are very rare.

Mysqli is generally short form of MySQL Improved for noting down. Now lets look into how we can use the mysqli library to connect and store data in MySQL. The general approaches that is used in mysqli are firstly connecting to the database, followed by inserting / updating / deleting / selecting records from the database and lastly closing the database connection. Mysqli also provides a function that can retrieve the last inserted id directly from the database often used for redirections after saving the record. Lets look into some of the important mysql functions and why they are used here which are in the procedural style.

- mysqli_connect(<host>, <username>, <password>, <dbname>, <port>) – This is used to connect with the MySQL server and selecting the database by passing the specific parameters as mentioned above. The parameters resembles MySQL host, MySQL user, MySQL password, Database name, MySQL port etc. This returns the database connection object often the variable in PHP is named as $conn.
- mysqli_query(<$conn>, <$query>) – This function is used to execute any of the queries by passing the connection object and the mysql query. For select queries generally a result object is returned while for other objects a bool value which is true in case there is some successful modification in the database or otherwise false. Also if in case there is some error while executing the query in that case too false is returned.
- mysqli_fetch_assoc(<$result>) / mysqli_fetch_array(<$result>) – This function is used after a select query and the result is converted to an associative (mysqli_fetch_assoc) / indexed (mysqli_fetch_array) array using it. This must be run in a loop as each record is retuned each time the function is called. Then the array retrieved can be sued to get the data from the records.
- mysqli_num_rows(<$result>) – This function is used to get the no of records returned in the result object using the select query and is often useful.
- mysqli_error(<$conn>) – We can use this function in case the other function does not execute successfully. In that case we can retrieve the error from this function passing the connection object and process the code accordingly.
- mysqli_real_escape_string(<$conn>, <$string>) – This is another very useful function in PHP and is used to sanitize the inputs before sending them to the database. This prevents multiple types of injections etc. The connection object and the string to be escaped are passed to the function and the return value can be safely stored into the database.
- mysqli_close(<$conn>) – This function is very vital as when we start a connection we must close it so that multiple connections does not stay open in the MySQL server at once which can lead to error for maximum allowed connections in MySQL in case the number of connections increases than the parameter value in the MySQL server. So we must close the connection by passing the connection object to the function as a good measure.

So these are many of the useful mysqli functions that are mostly used in PHP8 and onwards. To note there are many other functions available also in PHP8 that we can check when needed. For the purpose these are the mostly used and required functions while we process data in PHP to store in MySQL databases. Also these functions can be used in object oriented patterns as well which I am skipping here. For references to that you can check the internet though they are also useful. But for the purpose of the book and also I mostly prefer the procedural style of mysqli functions I am only mentioning the procedural functions here. So this concludes how we can use MySQL from PHP and we can check more of it later when we shall go over on how to write full fledged application using the stack mentioned in this book. So lets halt this for a while and look into some of the best practices we should employ while using MySQL standalone and also along with PHP.

The best practices we should use while writing MySQL in PHP should start with always using a try-catch block for MySQL functions often useful for runtime errors, use separate users for each databases with strong passwords as this can be often compromised, define separate functions in project which can be beneficial for better handling the queries and their result, using limits in queries most of the time as large result sets can often slow down the page loads, Also use checks before processing the results if the result set has records or even if the query executed successfully, employ proper error handling mechanisms, sanitize inputs as mentioned etc. So by keeping all of these things in mind we can move on to the next chapter and learn Smarty a bit before moving on and checking how to develop business critical applications in PHP, Smarty and MySQL.

BASICS OF SMARTY

Lets start discussing Smarty. So first of all lets get an idea of what is Smarty as most of us can be unaware of this stack in web development. Smarty in one line is a templating engine used to create front end templates for integrating with PHP. We shall dive deep and look into details of what it offers but firstly lets see basics and what actually Smarty is. So what is Smarty?

Smarty is primarily a templating engine as it is said and is available as a library developed in PHP. We can use this library to load and parse the template files as specified in the Smarty library. There are lots of specific things available for creating and working with Smarty templates we need to know. To brief for the Smarty templates there is some specific syntax also available along with we can use the PHP functions too within it. Besides to note only when first time a page is loaded the template is interpreted and cached while subsequent calls are from the cached files only which make it faster since then. Now lets look briefly into how Smarty is used with PHP including its caching mechanism. This will be followed by looking into the language syntax and its security mechanism that is available with it. You can go through any book that provides complete reference to Smarty as well for all its features. For our purpose the things we discussed would be good to proceed and look into developing apps using the stack.

As of now while the book is been written the Smarty version available is 5.x. We shall refer to this version for all our purpose in this book. So Smarty is used with PHP using the smarty library available that can be downloaded from their website or using git repo available for them. Once downloaded or installed that can be done in any project directory like libraries we need to include the smarty autoloader in our pages. This is followed by creating an object for the Smarty class which object we shall use in all our further references. This can be done by using the below syntax.

```
require(<path_to_smarty_library>/Smarty.class.php);

use Smarty\Smarty;

$smarty = new Smarty();
```

Instead of calling the Smarty class in our code we could have called the autoloader script also in case we have installed via composer. Now using these above lines of code we have instantiated the smarty class by creating the smarty object. Now we can use this object for rest of our purposes like assigning variables or loading the template files along with the variables. Now after that we can look to set the default template, compile, cache or other directories in which case we need to only pass the relative path to the smarty object later. We can do that as below. The setting of the directories is also done using the smarty object.

$smarty->setTemplateDir(<path_to_template_directory>);

$smarty->setCompileDir(<path_to_compile_directory>);

$smarty->setCacheDir(<path_to_cache_directory>);

…

Like this we can set the various paths like config etc as per requirement. Though most of the cases setting the template directory, compile directory and cache directory is sufficient to proceed. So once set we only need to pass the relative path to the template files for loading them in our scripts. The template directory is used for the location of the smarty template files while compile directory is used for the compiled files that are actually created after compilation and loaded subsequently while we load the PHP page. Lets look into the Smarty compiling and caching mechanism now.

Once we set the compile directory in Smarty that is used to store the compiled template files to be loaded in the PHP script where the template is loaded. Now when we first call the PHP page the template is compiled. The template files are basically tpl files or more accurately have .tpl extension. Now on first loading they are compiled to subsequent PHP files and those compiled PHP files are stored in compile directory. Now while next time the same page is loaded the PHP file from compile directory is loaded for the template and no compilation is required on run time. This makes the process a little bit faster.

Only if we make any changes to the template (.tpl) file in that case the next PHP page load recompiles the template file and replaces the earlier compiled file with the new compiled file. Subsequent calls load this new compiled file onwards. Hence for Smarty requests the template file compilation takes place for the first time only or in case of any changes to the template file. Subsequent calls uses the file that is already compiled. So this can be handled using compile check mechanism also that is available inbuilt but not very frequently required. Now lets look into the caching mechanism of Smarty.

By default caching is set to off in Smarty. To enable we need to set the below variable as mentioned.

$smarty->setCaching(Smarty::CACHING_LIFETIME_CURRENT);

$smarty->setCaching(Smarty::CACHING_LIFETIME_SAVED);

We can use either of those parameters while setting caching to on in Smarty. We can set it back to off using below lines of code.

$smarty->setCaching(Smarty::CACHING_OFF);

While we set the caching to on we can use caching_lifetime_current which uses the currently set cache time limit for the Smarty instance. We can set the cache life time to any value in seconds though its default value is 3600 seconds or 1 hour. We can use below syntax to set the cache lifetime.

$smarty->setCacheLifetime(<time_in_seconds>);

We can perform all this task in a config file that we can define while setting the directories. The default cache directory is cache inside the template directory but we can set it to any other location using the syntax as described above. Once cache is set the template files are loaded from the cache directory still its valid. The validity is checked using either the time limit or if the template (.tpl) file is modified. In both cases caching mechanism is used to recache the file. In other cases the template files are loaded from cache directory which makes the process even more faster. We can use cache check mechanism of smarty to check if a template file is already cached for specific purposes.

In case we require we can clear out the Smarty cache also. This is done using below syntax.

$smarty->clearCache(<template_file>); // clears specific file cache

$smarty->clearAllCache(); // clears all cached files.

$smarty->clearAllCache(<time_to_check>); // clears all cache previous to mentioned time

$smraty->clearAllCache(Smarty::CLEAR_EXPIRED); // clears caches that are expired

So using this mechanism we can create, call, compile, cache and execute the Smarty template files. By keeping this basic mechanism in mind we are good to proceed to our next section hoping we have very good concept now on how to use Smarty with PHP scripts. This would be very much beneficial while we shall look in a bit broader sense on using the library or the templating engine in the chapter where we shall look on how to create business critical projects using the stacks. So lets move on to our next section on the library, its setup and how we can pass the variables while calling the template files from our PHP scripts.

Smarty library itself is coded in PHP. Present version is 5.x which can be downloaded, installed using composer or by using git. The biggest advantage of using Smarty is that we can change the user interface when required only by simply changing

the template files or directory. There is no need to change the business logic and we can use the same passed variables in the new template files. We can use the Smarty object instantiated from the class to perform various operations supported by Smarty. The setup of Smarty we have already seen so we shall only see here some of the best practices regarding Smarty setup.

For best practices I always suggest to include and create the smarty object in a particular possibly config file and include this file in all those files where the templates needs to be loaded. This makes code readability and complexity simple and useful for develop complex business critical apps. This we shall look in the later chapter. Now lets move and look how to pass PHP variables from PHP pages to Smarty templates using the Smarty object.

We generally follow the flow as the variables are assigned to Smarty before loading the template files. This is generally done using specific functions. Lets look at the syntax below.

```
$variable = ‘’;            // initialize the variable

$smarty->assign(‘<smarty_var_name>’, $variable);

$smarty->display(‘<template_file>’);
```

Here firstly we have a PHP variable which is assigned a value. This PHP variable is assigned to the Smarty variable through the assign function. In the Smarty template we can access this variable using the name of Smarty variable specified here. This is followed by the template file that is loaded using the display function. We can load multiple template files and any file loaded below the Smarty variable assignment can use the Smarty variable within it. These are some of the basic practices we would require while working with Smarty. To know we can assign variables of any data type to Smarty to be used within the template files. So this ends this section and lets move on to the syntax and code structures we can be using in Smarty templates.

To start keep noted Smarty syntax is very similar to PHP and while we can use PHP functions itself in Smarty. We shall look into all of these now. In this section we shall cover topics like basic syntax, variables, conditional statements, operators, built in tags etc. Lets start with the basic syntax of Smarty. So below is a single line how Smarty variables are written in a template file to be displayed in front end.

```
<html_tag>{$smarty_var_name}</html_tag>
```

Here in the above line keep note on the $ sign used before the smarty variable. While assigning the variable in PHP page we simply pass the name while to access in the template we use the $ sign in front of it. Also the whole variable needs to be enclosed by curly braces and this all comprises the very basic and elementary syntax that is used in Smarty. Though actually this curly braces acts as delimiter and can be modified too. Now lets dive a bit deeper.

Lets look into the syntax a bit more like how to include files or assign variables directly in Smarty along with parameters needed for same. For that lets look into the below syntax.

{include file=”smarty_header_template.tpl” nocache}

{assign var=smarty_variable value=”any_value”}

{mailto address=”mailto_address@domain.tld”}

So above are some of the ways we can use the Smarty tag to set a variable by passing its value as attribute or include a file without its cache or even directly calling PHP functions and set their attribute value. Lets look into it. In the first line we have included another template in this template itself and also specified not to pick it up from cache which is though optional. In the second line we have declared and assigned a value to the variable. While in the third we have assigned directly a value to the PHP attribute which is also supported by Smarty. So these are some of the basic things we can do in Smarty using its tags which has a lot of variations too. For more references I would suggest to go through a book with complete reference of Smarty or go through their documentation. This is not presently in the needs of the book so we shall go into next part and shall look into more of these in the chapter where project development will be focussed upon.

Now lets look into how we can access variables in Smarty. Generally we pass variables from PHP into Smarty. But as a page displays a lots of data multiple variables needs to be passed. For convenience to this problem often better solution is to pass a single array with all the variables as keys for the page or template. Hence as a part we can pass to the template variables that can be arrays or of any data type. We shall look into how to access each of them below. Generally variables are accessed directly whereas arrays by index or key. Lets look below.

<html_tag>{$smarty_var_name}</html_tag>

<html_tag>{$smarty_array_name[index]}</html_tag>

<html_tag>{$smarty_array_name.arraykey}</html_tag>

So these are some of the basic ways by which variables in Smarty can be accessed. Generally arrays are executed in loop to get specific data for each index. While they can also be accessed by their keys as displayed above. There can be lots of variations like passing variables itself in indexes or some other which are not required for this book at this stage. So with this information we are good to look into how to write comments in Smarty.

Along with comments in this part we shall cover about literals also in respective of Smarty. To write comments we can use a syntax similar to below.

{* <comment> *}

To write a comment we can enclose the statement within asterisk (*) symbol as displayed above. So this will be treated as comment in Smarty. Now lets discuss about literals. Literals are something needs to be used when does not want the Smarty to compile or process that part of code. Its requirement can be clear from below example.

<script>function <fn_name>{ return <ret_val>; }</script> // don't need literal

<script>function <fn_name>{return <ret_val>;}</script> // need literal

In above lines of code the first function has a space after the curly braces hence it is not treated as a Smarty variable. So this will not be parsed by Smarty as variable but rather as javascript only. So this would not need to be in a literal tag. While the second function there are immediate keywords after the curly braces. So it not passed in literal tags the code can be treated as Smarty variable and can throw errors while execution. For this purpose we need literal tag for this and similar cases needs to be passed in literal tags. Lets look into how literal tags are specified now.

{literal}

    function <fn_name>{return <ret_val>;}</script>

{/literal}

This is how we can enclose the function in literal tags to skip Smarty parsing. Now lets move on and look into remaining parts that are variable modifiers, conditional statements, operators, built in tags, custom tags etc. Lets start with variable modifiers. Variable modifiers are something that can be used with variables within Smarty tags for a desired output. These variable modifiers are generally PHP functions or similar thing that are appended to the variable with a pipe (|) symbol for the expected functionality.

We can pass parameters also as per requirement to the modifiers for desired result. Lets look below into a variable used with a variable modifier along with its syntax.

<html_tag>{$smarty_variable_name|variable_modifier:parameters…}</html_tag

This is the basic syntax of using variable modifiers along with variables in Smarty. Generally we use the variable modifier after a pipe (|) symbol and pass the parameters after the colon (:) if required. This is how a variable is used with modifiers. Now lets name a few variable modifiers available in Smarty which will make the concept even easier. To name a few are count, upper, date_format etc. So these modifiers are readily available in Smarty and can be used for vivid purposes. Also to note we can use multiple modifiers on the same variable in the same line using multiple pipes (|). This is demonstrated below.

<html_tag>{$smarty_var_name|var_modifier1|var_modifier2}</html_tag>

This is how they are used and also can be used along side parameters if required. There are huge lot of such modifiers available in the Smarty library. To list all of them you can pick any book providing complete reference of Smarty or check out their documentation.

Now lets move on and look into operators in Smarty. Smarty generally supports most of the operators that PHP supports like the addition (+), subtraction (-), multiplication (*), division (/), equality check (==) etc. There is a lot bunch of operators available in Smarty. To check the complete list you can check out their documentation as well. Now lets look how these operators can be used in Smarty templates. For that lets look out the below syntax.

<html_tag>{$smarty_var1 operator $smarty_var2}</html_tag>

This is the basic syntax we can use for operators in Smarty tags. Also to note we can use multiple operators in the same line as per requirement as well. The usage of operators in Smarty is very simple and according to PHP usages indeed. Now lets move on to next section and check out the built in and condition tags available in Smarty.

There are whole lot of tags available in Smarty but among them the most important ones are the conditional and loop tags that would be very frequently used in the projects. Lets look into those tags available in Smarty. The tags are {for}, {foreach}, {if}, {elseif}, {else}, {include}, {assign} etc. Lets look into the condition based and loop tags here along with their usages.

So firstly lets start with the {if} tag available in Smarty. Below is the syntax that is used with the tag. There is a variant to this tag that is the {if}{elseif}{else} tag. These resembles their counterparts in PHP for their functionality. Lets look into the syntax of the {if}{elseif}{else} tags below.

```
{if $smarty_var == value1}

    <html_code1></html_code1>          // HTML block - One

{elseif $smarty_var == value2}

    <html_code2></html_code2>          // HTML block - Two

{else}

    <html_code3></html_code3>          // HTML block - Three

{/if}
```

To be noted that an {if} block must be corresponded with a closing {/if} block. While the other blocks {elseif} and {else} are optional and can be used as per requirements. Generally the flow is similar to PHP if-elseif-else statements where if the condition is true then block one will be displayed while otherwise it will check the elseif condition and display block two if that is true or otherwise the else block will be executed and block three will be displayed in html output. This is one of the commonly used tags in Smarty. Now lets look into the another commonly used tag that is the {for}, {foreach} tag in Smarty.

The {for} tag is in fact used as a loop and it also ends with corresponding {/for} tag. Lets firstly look into the syntax and after that we shall discuss the flow of the code.

```
{for $i=$start_val to $last_val max=$max_val}

    <html_tag>{$i}</html_tag>

{/for}
```

This is how the {for} tag is used in Smarty. We can iterate in the tag from a start value to the end value and optionally pass a max attribute resembling the loop to stop if the value of $i exceeds $last_val or $max_val number of iterations has been completed. In the block we can pick the value of $i which is iterated and display the same in the html block. We can do similar things with arrays to display the values corresponding to their indices or keys. So these are the two very important tags we would need in Smarty

templates. Besides them we have already checked out the {assign}, {include}, {literal} tags in our earlier section. Lets name some more tags that are important in Smarty. They can be {while}, {function}, {section}, {nocache}, {insert} etc. Lets look into the syntax of {while} tag below.

```
{while $i < $value}

    <html_tag>{$i}</html_tag>

{while}
```

In the above syntax of {while} tag $i is checked to be less than $value and till that the block inside the {while} tag will be executed and displayed otherwise it shall get out the loop and subsequent lines will start executing which is also very similar to that of PHP. So these are how the built in tags are used in Smarty and forms a very important part of the templating files as used often. For the other available tags you can check out a complete reference or official documentation of Smarty. Now lets move on to the next section that is custom tags in Smarty.

Custom tags are available as plugins though many of them already stay available with the Smarty library of latest versions. Lets look into some of them like {counter}, {eval}, {fetch}, {mailto} etc. We shall look here into {counter} and {fetch} only. So lets start with the {counter} tag. The syntax of {counter} tag is as below.

```
{counter start=0 skip=3}<br/>

{counter}<br/>

{counter}
```

So this is how {counter} tag is used where it will print out the html “0
3
6” in output. This is often useful in many computations or operations in PHP. Lets look into the other tag that we discussed the {fetch} tag.

Below is the syntax of {fetch} tag.

```
{fetch file=”<URL>” assign=’var_name’}

<html_tag>{$var_name}</html_tag>
```

In the above line of code we are fetching the file from the URL and assigning the contents to a variable $var_name which is displayed in the later block. This is very similar to the fetch() function available in PHP. So these are some of the custom tags

available in Smarty. We have listed a few more and there are lot more of them available which can also be easily checked in their documentation. So this concludes the section of tags that are very important part of Smarty.

Now lets discuss another very important feature of PHP that makes it very versatile. This is its ability to use PHP functions itself inside the Smarty tags. We can in fact use any of the PHP functions available inside the Smarty tags which makes the library even more useful and hence can be used for almost all purposes. By keeping this in mind lets move on to the last section of this chapter that is the security mechanism available in Smarty. Lets look into it now.

In Smarty security measures are taken in two levels – template level and directory level. Template level security complies to various restrictions that can be applied for parsing the template files. This is generally enabled using the below function.

```
$smarty->enableSecurity();
```

This is how to enable security and in which we can pass an optional attribute security class which can modify the default security measures in Smarty. This handles PHP functions that can be used in template along with modifiers or PHP tags etc. This gives complete control over how the template should act while compilation. Though generally not very frequently used it may be essential for complex tasks. Similarly this can be disabled using the function disableSecurity().

Now about the directory level security that must be handled by a .htaccess file if not we are placing the template files outside document root. To disable direct access of the template files (.tpl) we can specify the same in .htaccess using the below lines of code.

```
<FilesMatch "\.tpl$">

    Order Allow, Deny

    Deny from all

</FilesMatch>
```

Though these lines of code can restrict template files access directly we can also specify every other rule along with this in the .htaccess required in the project. We shall look into that in later chapters. So this concludes our discussion with basics of Smarty. We shall in next chapter look into REST APIs before moving on to creating business level applications using the stacks.

BASICS OF REST APIs

So here we start our topic discussing about REST APIs. So before beginning lets look into the term what actually REST APIs stands for? The full form of REST API is Representational State Transfer Application Programming Interface. The concept of REST is related to stateful or stateless nature which we shall check later. Now for purpose to understand what actually a REST API is we can say that it is a resource in a web server which accepts some request and returns some response while hiding beneath the logic behind the response. Lets look into it a bit in detail.

Why actually a REST API is important. The main reason to use REST APIs is that it takes a request from the application server and returns the server without the necessity for the application server to know what logic it actually used to perform the task. It hence makes the complex business critical applications quite simple to handle using a layered approach. This is why they have been very popular in recent days along with the capability to reuse them by different application servers requiring the same task. That is why I prefer using this approach for creating application development. Lets look into it a bit more.

Lets first understand what is statelessness and layered approach in context of REST APIs. A state of an entity is its value which keeps on changing one after other based on previous value. But for REST APIs the state of an entity is not maintained hence each time the endpoint or resource is accessed the entity is only in its initial state. This property of REST APIs is called statelessness and they comply with the same. Hence state is not maintained for REST APIs. Again layered approach is something that makes the underlying mechanism hidden and only provide the result to the requester. This is also a property of REST APIs which make them very good entity to be worked with. Along with their reusability and secure nature make them very important in the world of application development. Now lets start with what actually comprises or are the components of REST APIs.

Components of REST APIs constitutes of those things that are required for the whole process to complete. Generally the major components of REST APIs are listed below.

- HTTP Method
- Authentication Mechanism
- Request
- Response

These four entities comprises of any REST API mechanism. Lets look into each of them in detail starting with HTTP Method. A HTTP method is a very important part of any API request and comprises the way in which the payload is sent over the network to the API server. Lets look into it now in a bit more detail.

HTTP methods are a way by which the payload is sent over the network to API server for processing. This is complied by the HTTP protocol for transferring data. Though there are a number of supported methods for transferring data over the network we shall see in best practices only few can be handful for performing any of the complex tasks. Lets look into the methods firstly below.

- GET
- POST
- PUT
- PATCH
- DELETE

Though these are the five methods set by standards for different purposes we shall look into what actually they are, how they work and when they are needed now. So lets begin with the GET method. GET method is a less secure method for REST API requests which can always be used while less secure or non binary data is being transferred. As per standards it is used for retrieving data from the server that can be unit or a cluster. In a GET method the payload is sent unencrypted and with the URL string which make it very unsecure hence data like passwords etc are not suitable to be sent using this method. In this method data is sent as query parameters to the REST API server. Lets look below at an example to a GET request.

GET <API_SERVER_URL>/<REQUEST_URI>?<PARAMETER=VALUE&…>

This is how a GET API request looks like. Mostly in PHP we make an REST API request using the curl library. For GET requests we can use similar URL with query parameter or passing them in payload and mentioning the method as GET in curl. Both the approaches actually works. Now lets look into some other features that accompanies with a GET request. Hence GET is generally used when insecure payload would be sent along with for retrieving results. This can be used without a request body. The response from API server for GET requests can be with any of the HTTP status codes as per the result. For developers we shall always look to properly test the APIs for security and functionality before going public. Now lets look into next HTTP method which is POST which obviously is most commonly used one.

In POST HTTP method data is sent over the network to the API end point in the body and hence not visible externally as in the URI for GET method. In POST method vivid data types can be sent over in the payload like binary, json etc. As in POST data is sent over body and not visible externally it is very secure mode and so data like password etc can be sent over using this method. Below is the format of sending data as in most of the APIs some authentication mechanism is applied before accessing the endpoints.

<AUTH_HEADER>

POST <API_SERVER_URL>/<REQUEST_URI>

<PAYLOAD_BODY>

This is a general format used while hitting endpoints using POST. Firstly we can authenticate the request using authentication headers. This is followed by the request with the payload that is processed in the API server and response is returned. POST method is used for creating or inserting new records in API server. But practically we can use POST for any of the operations and is because it is one of the secure methods. In curl as usual we can pass the headers, endpoint and data as parameters and retrieve the response to further process it in PHP. The response status code can be any of the available as depending on how the request behaved in the server. Now lets move on to next HTTP method which is PUT.

PUT method is generally used for updating records in API server. Though occasionally used by few developers it is the standard as mentioned to be used for updating or modifying records. It acts similarly as POST. Lets look into its syntax below.

PUT <API_SERVER_URL>/<REQUEST_URI>

<PAYLOAD_BODY>

In this method as for POST headers, endpoint and payload are passed. While once the headers are authenticated the payload is processed at the endpoint to return proper response and status code from the API server. In it also the payload is invisible externally and hence secure so can be used for any type of data in payload. Along with any datatype like binary, boolean etc are supported in payload for PUT. It is used in curl for PHP in a similar fashion as POST. PUT is specifically used for complete modification of the data in API server hence accompanied by payload having all the respective fields. Now lets move on to the next method that is PATCH.

PATCH method is also used for modification of data in API server but unlike PUT it only performs partial modification. All other things related to using with curl, usable data types, security or process is similar to PUT. This only differs according to the standard as it should be used for partially updating records in the API server. Now lets look into its syntax below.

PATCH <API_SERVER_URL>/<REQUEST_URI>

<PAYLOAD_BODY>

So this is how PATCH is used. Now lets look into the last HTTP method used for REST APIs. The method is DELETE.

DELETE is a HTTP method that is used for deleting specific records from the API server. Like updating deleting records can also be on mass data or single data either of them. Generally we need to pass the filter for data in both PUT, PATCH and DELETE. The filter can be a unique id as for single deletion or a specific filter for DELETION of set of data. Unlike PUT and PATCH for DELETE we generally do not need any other payload to be sent as its clear from the operation. The unique id can be sent as payload or even in URL which ever is suitable. In DELETE requests also we need to send the headers to authenticate, endpoint and payloads to process the request. On which completion a response and a status code is returned from the server. This can be easily accomplished using curl in PHP similarly as we seen earlier. Below is an example of a DELETE method.

DELETE <API_SERVER_URL>/<REQUEST_URI>/<IDENTIFIER>

So this is how DELETE method works in context of REST APIs. So we have gone through all the available methods that can be used for REST APIs. Though among all these we shall generally be using GET and POST methods mostly as they are convenient and can be standardized for any of our tasks. Hence this concludes the section of methods that are used with REST APIs for API requests. Now in next section we shall look into the various authentication mechanisms available to secure REST APIs along with some of the libraries that can be used to implement them.

Authentication mechanisms are ways to secure a REST APIs from unauthorized access and provide a security mechanism for its intended users implemented by security keys. There are different types of security mechanisms that can be implemented from simple to very complex which are difficult to breach. This extends from simple keys to multi level key pairs, restricted accesses etc. Lets list down the major authentication mechanisms that can be used with REST APIs.

- API Keys
- HTTP Basic Auth
- OAuth 2.0
- JWT

The above listed are some of the majorly used authentication methods which we shall look here. There are many other mechanisms which can be studied over the web as intended. But before we study the mentioned methods lets look into some important concept that can be beneficial to understand them like CORS, token rotation, rate limits etc. Firstly CORS stands for Cross Origin Resource Sharing. It is used to limit the API

server to be accessed by specific client domains only and is a preliminary method for API security. It can be used along with the methods for best usage. For implementing CORS we can use some below architecture in API server side.

Set Access-Control-Allow-Credentials: true

Set Access-Control-Allow-Origin: <https://intendedorigin.tld> // Wildcard (*) not accepted for allow credentials

Set Access-Control-Allow-Headers: allow

By using above set of rule we can implement CORS in our REST API architecture. Except that we can use features like token rotation as well. This is a good way to protect the APIs from token breaches and can be implemented by multiple ways from manual to automated. And thirdly rate limiting is another feature to prevent the API server from DDoS or mass requests. This can limit each IPs to limited number of requests in a certain timeframe. Rate Limiting can also be implemented by multiple different mechanisms. So by keeping these things in mind we can start with the API Authentication methods and look how they work. We shall start with using API keys.

API Keys are generally a mechanism where each API is authenticated by a specific or a set of keys. These keys are authenticated on the API server side to verify if the request is legitimate and only upon successful verification the server processes the request. The API key must be send over the header from the requesting server which is validated at the API server. These keys should be unique and are shared as a public or one key per user basis for keeping count of the requestor. This helps in applying limits based on the keys. Below is how an API key can sent over the header.

curl <https://endpoint.tld>

-H “Authorization: Bearer <API_KEY>”

By above manner the API key can be sent over to the API server for successful validation. It is secure but lacks the capability for user specific authentications. Now lets look into the next method of authorization which is HTTP Basic Auth.

In HTTP Basic Auth the authorization is mostly done using a username and password pair. It is similar to API keys in some ways but it differs mostly in the way the secret is sent. Firstly the username and password are appended using a colon(:) which is followed by base64 encoding of the same before sending it in the header to the API server. In the API server it is decoded and the username and password are extracted to be verified. On only successful verification the API server processes the request. Lets now look into

how the headers are sent using curl for a username – password pair of admin – admin. The mechanism is as described as above which is firstly the pair will be appended as “admin:admin” for encoding the same. This is followed by base64 encoding of the username – password pair say to “YfHhgGh655fCg67gt==”. Now this encoded string is sent over headers as below.

```
curl <https://endpoint.tld>

    -H “Authorization: Basic <base64_encoded_string>”
```

Once send over to the API server they are decrypted and validated for authorization. Curl in PHP has very powerful libraries to manage the whole in both the requesting server and API server end which we shall look later. Another advantage of HTTP Basic Auth is it can be used for user specific accesses as well hence making it suitable for dynamic platforms too. Below is a code for using HTTP Basic Auth in PHP.

```
$ch = curl_init(<endpoint_url>);

curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);

curl_setopt($ch, CURLOPT_USERPWD, “<username>:<password>”);

$headers = [

    <!— “Authorization: Bearer <api_key>” --!>          // for API Keys

    “Content-type: application/json”,

    “Accept: application/json”

];

curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

$response = curl_exec($ch);                                  // receives the response
```

So this how a API request can be made using API keys or HTTP Basic Auth in PHP using the curl library. Now lets move on and look into the next most important authorization mechanism used in PHP which is the OAuth 2.0 mechanism.

OAuth 2.0 is one of the modern and most secure mechanism for authenticating REST APIs. This can be used for public APIs along with user specific APIs as it provides a mechanism for user authorization as well. To understand OAuth 2.0 better lets firstly look into its basic mechanism. Basically a username and password is provided to the user to access the REST API server. While the requestor sends the credentials it is validated at the API server end and a token is generated for the connection which is shared to the requestor in response from the API server. In subsequent accesses the requestor can use this token to authenticate their requests to the API server. There is also a mechanism which allows the token to be refreshed after a specific expiry date and time. Once the previous token is expired a new token is shared in a similar manner and this shall only be valid in subsequent requests and whole refresh mechanism is carried out without need to use to the login credentials. So this is how OAuth 2.0 mechanism works. It is basically a mechanism or concept that can be implemented in multiple ways by any REST API development team. Lets look into the flow once diagrammatically below.

Requestor (User / Password) → API Server (Authenticates) → API Server (Shares Token) → Requestor (Receives Token) → Requestor (Token) → API Server (Verifies Token) → API Server (Processes Request) → API Server (Shares Response) → Requestor (Receives Response)

So above is the whole process by which the OAuth 2.0 mechanism works. Now for reference there is a library for OAuth 2.0 built by our end that makes the whole process easier and far more secure. We shall be referencing that library in our subsequent chapters while building real time applications. Now just note the URL to give a look into the library names SARK Auth. Below is the link.

GitHub Link: https://github.com/shouvikkrsarkar/sark_auth

So this is what we need to know for now about OAuth 2.0. Now lets move on to another way of securing REST API servers which is mostly introduced after the rise of the JavaScript frameworks for front end but can be equally used with PHP too. This is called JWT or JSON Web Tokens.

So what is a JSON Web Token. Firstly we can note that JWT can be used for stateless REST API development which means the API server need not require to store user credentials in session etc rather each time its in its initial state and the credentials can be received each time a request is made for specific response. So how it works. Firstly a request is made from the requestor with user credentials which is validated at API end which generates a token for the user. But this token is further encrypted using algorithms like HS256 or RS256 and shared to the requestor in front end. Each time the requestor needs to send this token to get access and the request be processed. Each token

should have access to resources specific to the user with the credentials only. This part of the workflow is similar to OAuth 2.0 somewhat. But the major difference is that the token is encrypted and is because it needs to be stored on front end and can be available from browser. The token can be stored as a secure cookie or in local storage. So this token must be destroyed once the user signs off or it should have a expiration date and time to prevent its unauthorized usage. These JWTs are mostly used with front end JavaScript technologies like React JS or Angular JS but can be used along with PHP as well for beneficial purposes. So this is some of the mechanisms how user authentication works for REST APIs mostly in context of PHP. Now we shall limit our discussion on this and lets move on the getting a broader idea of the Request and Response concepts for REST APIs.

So lets dive deep into the request format for an API request. Generally an API request consists of three objects. They are namely endpoint, header and payload. Lets look in brief what actually they are?

An endpoint is the resource URL where the request will get processed. It generally comprise to a page in the API server and more specifically the URL to access it. The URL can be used as per convenience using files like .htaccess but majorly is directed to the PHP page in the API server which processes it. Besides headers are also an important part of an API request. In headers we need to pass specific parameters that is identified in the API server which includes the request data type eg. JSON, origin which is sometimes essential when CORS has been used, authentication credentials or keys which is used for authenticate the request and many other parameters. Sometimes for chunking specific other headers would also need to be sent over. These headers actually is the layer which prevents the payload to reach or processed before actually getting an idea of it. The third and important object is the payload which itself consists of the data that needs to be processed. This data is processed differently for different cases and a response is returned after process is completed. The data can contain any data type which can be converted into a single string of JSON which is efficient while transferring data over the network to the REST API server. The response also mostly is returned in JSON format which is further decoded to array when received.

So the request of an API server is as explained above. Now lets look into how the response from the API server should look and its major constituents are formed by HTTP response code and response body. Lets dive deeper into it.

Basically a API server response comprises mainly of the response along with the HTTP status code which identifies the message type. But to formulate the response firstly it needs to be identified that what data format to be used for the response. The mostly common data type is JSON as for request which stringifies and can be constructed back later. The JSON can be decoded in the request server once its received

back. Along side the response data the HTTP status code is another important entity in the response. For successful responses generally 200 OK is returned from the server. Lets now look into the various major response status codes and which specific scenario they can be used. Lets look into that below.

- 200 OK – This is the standard success response code and we should be using this in most of our successful responses.
- 201 Created – This is used when a new record is created viz insert operation in the API server.
- 204 No Content – When there is no response 204 can be returned.
- 301 Moved Permanently – These are server specific responses and is returned when the endpoint has moved permanently to some new URL.
- 302 Found – This is returned when the URL is temporary available and later can be changed back.
- 304 Not Modified – This status code can be used when the response data is no different from the last one.
- 400 Bad Request – When the request body is malformed then in such case a bad request can be returned.
- 401 Unauthorized – This is returned when a request with authentication fails. In that case the access is unauthorized.
- 403 Forbidden – When a resource in the API server should not be accessible then trying to access it should give a forbidden status code.
- 404 Not Found – When the endpoint is not existing for such a case a 404 status code can be sent.
- 429 Too Many Requests – If there is considerable requests in short time mostly over a rate limited API server then status code 429 can be returned.
- 500 Internal Server Error – This status code is returned in case there is some error occurs in processing which was not handled properly.
- 502 Bad Gateway – This is returned if the endpoint for the request is improper and hence is not available to process the request.
- 503 Service Unavailable – When the API server is down or is under some server level error then this response code is suitable.
- 504 Gateway Timeout – When the API server times out before sending a response like max execution time then 504 status code is proper.

So we have listed above the major status codes that are sent over as a response from the API server. Among them 2xx are success codes, 3xx are temporary endpoint errors, 4xx are endpoint errors and 5xx are server errors.

By keeping these in mind it would be easier to go further and discuss developing APIs in our next chapter “Creating a Sample Project”. So as we have gone through major topics for understanding and developing REST APIs we can now move on to the

next chapter. In our next chapter we shall be using what we discussed over PHP, MySQL, Smarty, configuration and REST APIs combine them together to discuss further on “Creating a Sample Project”. Along side this we shall focus on system design which is a major part is creating efficient real time software applications. By combining all these after the next chapter we can be having good hands on over how to create real time web apps. So on this note lets end this chapter and move on to the next one and study on creating applications using PHP.

A SAMPLE PROJECT

So lets start our new chapter “A Sample Project”. Here we shall apply all over technical stuff we have studied along with system design to create real time web applications. System design is basically a very important part for creating complex real time applications. Here we shall be using a REST API and an application separately in which also the application will have layers between front end and business logic. This all would form a part of system design. For database we shall be using MySQL. Here we shall look over a very useful system that makes development of applications easier and maintainable and focus on that. For your projects you can surely use this design, some other or can create your own. Definitely there can be lots of types of system designs that can be good enough but the one we will be using here is surely one of those. So lets start our discussion. Lets take an example of URL Shortening project for reference.

To start development in local environment the very first thing needs to be done is supposed to be setting up the hosts file and virtual hosts in Apache for a real server experience which makes it more easier to develop with context of real world. For any application we generally need three projects firstly the application, secondly its API and third an admin panel for the application. So lets look into configuring the hosts first. This consists of configuring the domain in hosts file and httpd-vhosts.conf file. Firstly lets look into the hosts file.

The hosts file is located in C:\Windows\System32\drivers\etc directory by default. We need to open the hosts file with administrator privileges and edit it as by inserting the below line of code for all the entries.

```
127.0.0.1       domain.tld
```

This will redirect the domain.tld url to 127.0.0.1 or localhost. Next we required to identify the domain.tld and route the request to be served from specific directory in the Apache public directory instead of the root public directory itself. This can be done from the httpd-vhosts.conf file. This file is located inside C:\xampp\apache\conf\extra directory by default for xampp installation. We need to write the below code in the file for enabling both HTTP and HTTPS access for the domain.

```
<VirtualHost 127.0.0.1:80>

    DocumentRoot <path_to_apache_root_directory>/<project_directory>

    ServerName domain.tld

</VirtualHost>
```

Below line of code is for HTTPS access which would also be needful while we code.

```
<VirtualHost 127.0.0.1:443>
    DocumentRoot <path_to_apache_root_directory>/<project_directory>
    ServerName domain.tld
    SSLEngine on
    SSLCertificateFile <path_to_ssl_cert_file>
    SSLCertificateKeyFile <path_to_ssl_cert_key_file>
</VirtualHost>
```

So using the above code we have now setup the domain for one of the applications. We need to do the same for the api and admin domains as well which would be on a similar manner. After this I would suggest to install MySQL 8.0 in port 3306. While running xampp control panel you can start the Apache server and MySQL would run as a service and can be accessed using PhpMyAdmin tool. Another software I recommended is to install HMailserver and configure for domain.tld followed by creating some dummy email addresses for testing email functionalities. All the links to guide on how to is provided in first chapter and can be rechecked for proper setup before continuing. So in this manner we have three document root directories for three domain.tld specific to application, api and the admin panel. Now we are good to go and can have a look into the system design and software code which would obviously be inside these document root directories. Before proceeding I would suggest to test the domain.tld URL using both HTTP and HTTPS to confirm everything is setup correctly. This should be proceeded by opening MySQL using PhpMyAdmin and create a database by name of the project or any convenient with proper engine. Once done and domain.tld checked and hmailserver configured we are good to go to write some code. So lets look into that now.

The very first file that needs to be created in any of the project directories would be the .htaccess file. So using that preliminarily we can protect the directories and some specific files from direct access using URL from the web server. This can be done using below lines of code in the .htaccess file. These lines creates the basic redirection the server root and protecting specific files and directories from unauthorized access. Once this is done we can next write the code for accessing specific files in the directory with specific URLs as required which makes the server more secure and handy. Here we are protecting directories and file like htaccess, tpl (smarty template), etpl (email template), log (logs), xml (default values), sql (database) and config.php file which. We shall look later that other PHP files can be protected using specific code that we use in config.php

file which itself can not be protected using the same code. So we have to do the same using the .htaccess file. This config.php files contains all the configuration related to the project and can be used in any other files that requires it. This all setup is related to the specific system design I am mentioning here and as already mentioned for different system design patterns other structure can also be used.

```
RewriteEngine on


<FilesMatch "\.(htaccess|tpl|etpl|log|xml|sql)|config.php$">
    Order Allow,Deny
    Deny from all
</FilesMatch>


Options -Indexes


RewriteCond %{REQUEST_URI}::$1 ^(.*?/)(.*)::\2$
RewriteRule ^(.*)$ - [E=BASE:%1]


RewriteCond %{REQUEST_FILENAME} -f [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule .* - [L,QSA]
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
```

Hence this is the default .htaccess code and should be used in all the three root directories respective to application, API and admin panel. Now we shall look into the

folder structure that can be implemented for the projects which make the code easy to read and write, secure and very much organized for possible future additions or modifications. Lets look into that now.

Lets look into the structure of the application firstly. But before we move on lets look into the code that we can use in every PHP file to prevent their direct browser access. This should be used in the files that would be included or through AJAX rather be directly accessed. For this we can define a access key in config.php. This can be later checked in respective files to see if the access is from specific directly accessible files which have config.php included or being made directly using the URL. So below is the code for included and accessed through ajax files.

For included files we can use below code.

```
if (!isset($access_key) or $access_key != '<access_key_val>') {

    // Raise a 404 Not Found error and exit the code.

}
```

For files accessed through Ajax below code would be useful.

```
if ((!isset($access_key) or $access_key != '<access_key_val>') and (!isset($_POST) or empty($_POST))) {

    // Raise a 404 Not Found error and exit the code.

}
```

For files in Ajax in which $_FILES also needs to be checked which we shall look later. Now lets move on and look into the default structure of the application directory which will make understand the system design more conveniently.

```
project_dir

    assets

        theme_dir

            <!—HTML/CSS/JS Files From HTML Themes--!>

    config
```

config.php

functions.php

values.xml

includes

ajax

<!—PHP Files accessed from AJAX--!>

essentials

site_functions.php

authorize.php

libraries

smarty

<!—Other Useful Third Party Libraries--!>

resources

default

<!—Default Images--!>

logo

<!—Project Logos--!>

support_files

css

custom_style.css

custom_style_mobile.css

js

pages

<!—JS files for each page--!>

js_functions.js

ready.js

<!—Other Requisite JS files--!>

themes

email_templates

<!—Email Template Files--!>

smarty_template_dir

includes

<!—Smarty Application Header File--!>

<!—Smarty Page Specific Template Files--!>

uploads

<!—User Specifc Uploads Made Through Application--!>

.htaccess

<!—PHP Files Accessible Through URL--!>

So now we shall look into details of each of the usage of the files that is required and mentioned in the application directory structure. We shall follow that by understanding the working of SARK Auth the OAuth 2.0 implementation library we use and after that we shall look into the directory structure of the API portal and Admin panels. So lets begin with understanding the structure of the Application directory.

Inside the foremost project directory there is the assets directory. It consists of the theme files including CSS, JS and vendor which would be required for the UI for the application. These files are generally included in the smarty template files mostly in the header and footer and as per requirements. This folder can also contain vendor JS or

CSS files which would be required in the application UI besides the base theme files that we would be using for the application.

This is followed by very important directory config containing the config.php and other major files (functions.php and values.xml) which makes the backbone of the application and is used in every page we execute over the URL. We include the files and their constitutes in the config.php file and call the config.php file in every our php page in the application. Lets look into what constitutes these files starting values.xml.

So the values.xml file is an XML file in proper XML format which contains the overall site specific values. This file is protected by .htaccess file. It contains values for database connection, base url, SEO specific, default emails and various keys. They are stored in XML format which is accessed from the config.php file and then can be passed on to every PHP page and template files. Lets look into the dummy structure of values.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>

<settings>

  <database>

    <db_host>…</db_host>

    …

  </database>

  <root>

    <base_url>…</base_url>

    …

  </root>

…

</settings>
```

This is the default structure followed by the file and various other parameters needed can be added along with this dummy as per your requirement. The files is accessed in config.php as below.

```
$values = simplexml_load_file($base_dir.'config/values.xml');

$db_values = json_decode(json_encode($values), true)['database'];

$host = $db_values['db_host'];

$base_values = json_decode(json_encode($values), true)['root'];

$base_url = $base_values['base_url'];
```

Similarly any other values can be fetched from the values.xml file and can be passed into any other respective PHP page or template files. Now we shall look into the functions.php file which will be followed by the config.php file itself.

The file functions.php is a file which can contain multiple classes and should contain various functions that would be commonly required in various PHP pages used in the application. This is included in config.php file and hence available to all the PHP pages. Lets look into a dummy structure of the file. This file would be protected using the access key method we discussed earlier.

```
class database_functions

{

    function database_connect(<paramerters>){

      // function body

    }

    …

}
```

As database operations are very common and important we can always look to create a separate class for the operations. The other class main_functions I generally use can contain every other useful common functions like authentication, mail, calling API or any other that would be useful. These functions are available throughout the application code. The most important is the call API function which we shall look here in brief that actually how that works. This functions is in correspondence with SARK Auth which we use with the API authentication. Lets look into these below. Below is the dummy code for the main_functions class.

```
class main_functions
{
   public function __construct($userid, $authkey)
   {
      $this -> userid = $userid;
      $this -> authkey = $authkey;
   }
    function access_api($url, $data, $file='')
   {
      $url = $api_url.$url;
      $access_key = <access_key>;
      $default_secret = <default_secret>;
      $secret = array(
            'first_secret' => <first_secret>,
            'second_secret' =>  <second_secret>,
            'third_secret' => <third_secret>
         );
      if (!isset($file) or empty($file))
      {
         $data_to_send = array(
               'secret' => $secret,
               'user_id' => base64_encode($this -> userid),
```

```
            'user_secret' => $this -> authkey,
            'data' => $data,
        );
}
else if (isset($file) and !empty($file))
{
    $data_to_send = array(
            'secret' => $secret,
            'user_id' => base64_encode($this -> userid),
            'user_secret' => $this -> authkey,
            'data' => $data,
            'file' => array (
                    'file' => $file,
                    'extension' => pathinfo($file, PATHINFO_EXTENSION)
                )
        );
}
$data_to_send = http_build_query($data_to_send);
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_setopt($ch, CURLOPT_USERPWD, $access_key . ":" . $default_secret);
```

```
        curl_setopt($ch, CURLOPT_TIMEOUT, 30);
        curl_setopt($ch, CURLOPT_PROXYPORT, 3128);
        curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
        curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
        curl_setopt($ch, CURLOPT_POST, 1);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $data_to_send);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
        $response = curl_exec($ch);
        curl_close($ch);
        $result = json_decode($response, true);
        return $result;
    }
    …
}
```

So this is the structure of the class with the function access_api has been provided in detail. We can define any other functions that would be essential in the application here as per requirement. Now we shall look into how actually this function access_api works in correspondence with SARK Auth in the API server for calling which this is used. To note that for calling the function access_api we need variables $user_id and $auth_key which are passed to the constructor of the class. Lets look into that now. This would be in accordance with understanding the SARK Auth library.

First lets look how to set up the SARK Auth for your API directory. You can download the SARK Auth from GitHub from the link provided and firstly import the SQL file provided along with it. It would create an table api_authorization in which data is created while an user creates his account. This table is used only when user specific authentication needs to be implemented. We shall look into those codes later. For now lets look into how we set up and use the library. Next we need to replace all the keys values in authenticate.php file with any value of your choice. Once this is done we need

to generate the secrets and access keys to access the APIs. This would be done using below code which can be written in a authenticate.php file and including it in all the PHP pages that are to be accessed in the API server.

```
require_once __BASE_PATH__.'resources/libraries/sark_auth/autoloader.php';

use \SarkAuth\Basic;

$authenticator = new Basic\Autheticate();

$access_key = $authenticator -> get_access_key();      // retrieve access key

$secret = $authenticator -> get_secret();              // retrieve secrets
```

By using the above code in a file we can get the $access_key and $secret (array with default, first, second and third secret) which gives us the various values. We need to store these values for later usage.

To implement the authentication we need to use this below code.

```
require_once __BASE_PATH__.'resources/libraries/sark_auth/autoloader.php';

use \SarkAuth\Basic;

$authenticator = new Basic\Autheticate();

if(!isset($_SERVER['PHP_AUTH_USER'])or!isset($_SERVER['PHP_AUTH_PW'])
)

{

    $authenticator -> stop_exec();

}

$access_key = $_SERVER['PHP_AUTH_USER'];

$default_secret = $_SERVER['PHP_AUTH_PW'];

if ($authenticator -> verify_credentials($access_key, $default_secret) == false)

{

   $authenticator -> stop_exec();
```

```
}

if (!isset($_POST['secret']))

{

    $authenticator -> stop_exec();

}

if (count($_POST['secret']) != 3 or !isset($_POST['secret']['first_secret']) or !isset($_POST['secret']['second_secret']) or !isset($_POST['secret']['third_secret']))

{

    $authenticator -> stop_exec();

}

$first_secret = $_POST['secret']['first_secret'];

$second_secret = $_POST['secret']['second_secret'];

$third_secret = $_POST['secret']['third_secret'];

if ($authenticator -> authorize($first_secret, $second_secret, $third_secret) == false)

{

    $authenticator -> stop_exec();

}
```

This will authorize the API request at API end. Though this mechanism is for non logged in users. For users there is another separate mechanism we need to implement. After that we shall look into how to access the API from curl code in the application end. But firstly lets look into the user authenticate mechanism. For that in correspondence to the above code we need to implement the below code in a separate file and include this file in those pages where user specific data is to be accessed. This all can be done in a user_authenticate,php file in API end which we shall look later on.

```
$user_authenticator = new Basic\User_specific_autheticate();

if (!isset($_POST['user_id']))
```

```
{
    $authenticator -> stop_exec();
}
if (!isset($_POST['user_secret']))
{
    $authenticator -> stop_exec();
}
$user_id = $_POST['user_id'];
$user_secret = $_POST['user_secret'];
if ($user_authenticator -> verify_credentials($conn, $user_id, $user_secret) == false)
{
    $authenticator -> stop_exec();
}
```

This will check the user_secret with the user_id and verify if its authorized. To generate the user_secret we need the below code while user creates an account in the application.

```
$sql = 'insert into api_authorization(user_id, secret, active) values('.$userid.', '".$api_token."', '.$active.')';
$this -> database_func -> insert_into_database($conn, $sql);
```

This SQL statement will insert the secret or user secret in the api_authorization table. This would be used each time users logs in then this would be fetched and sent to the application and application sends it back with each request to verify with the user id at API end if the request is from the legitimate user. Lets look into the code for sharing of the secret with application from Api while user logs in below.

```
$auth_key = $user_authenticator -> get_secret($conn, $userid);
```

This should be used for the setup required for user authenticator as we seen earlier. So by this method we can retrieve the user secret from the saved data into the $auth_key variable. This data can be now shared to the application server in the response from the API server where it would be stored in session for further process of accessing the user specific data from the API server. So this explains how we can use the SARK Auth in API end for proper authorize and authenticate requests while implementing OAuth 2.0 mechanism. Now lets look how we need to use curl in our application side code to access the API server with proper access rights. Lets look into that now.

So as we looked we have received or retrieved the access key and secret for the API server. Now we simply need to pass these security credentials through curl. Firstly lets look into the code that can be used and later we shall look into how it works. We can create a function and keep it in functions.php file so that it can be used throughout the application. This code is written in earlier part of this chapter and the function named access_api which now we shall look into the mechanism. To study the code the above section can be checked.

Generally there are three parameters that the function access_api accepts which are endpoint, data and file. The file parameter is optional by default. We specify the various keys in the variables in the function. Also to note the user_id and auth_key for user are send in each request. If the user is not logged in hence the parameters are empty they are send empty only in POST body. Now access_key and default_secret are sent over the network as headers, while other secrets are sent over POST body. In case user is logged in the user_id and auth_key are received through the constructor into the class from config.php while creating an object of the class and sent over the POST body. The user_id is base64 encoded. All these along with the data and file if there is are sent over through curl request and received in the API server. The API server processes the whole request including authentication and processing the data and returns respective response which is further received in curl response and returned from the function and processed further. So this altogether how the SARK Auth authentication works in accordance with API server and how to send request and receive response from the API server through implementing a good secure Application API communication mechanism. Now after looking into the access_api mechanism and functions.php file lets move on to most important file of the application part which is the config.php file being included in every other PHP file in the application directory.

The config.php file is the most important file and starting point of code for any of the PHP pages. This file mainly defines the access key, base directory and other variables along with initializing the smarty object and others required. Lets look into some of the important lines of code written in this file and shall be followed by understanding the lines.

```
$access_key = ‘<any_static_value>’;

…

error_reporting(1);

date_default_timezone_set("UTC");

session_start();

$base_dir = '<path_to_application_directory>';

…

$values = simplexml_load_file($base_dir.'<path_to_values.xml_file>');

…

$userid = $_SESSION['userid'];

$auth_key = $_SESSION['auth_key'];

require_once('functions.php');

$database_func = new database_functions();

$main_func = new main_functions($userid, $auth_key);

…

require_once(SMARTY_DIR . 'Smarty.class.php');

$smarty = new Smarty();

…
```

So these are the major lines of code of config.php file. Firstly we define the $access_key which will be required to authorize and prevent direct access of any other PHP files that should not be accessed by URL. Then we set the PHP config values like error_reporting() etc. This is followed by the most important defining the $base_dir and later we can reference any file based on this. It is followed by getting the values from the values.xml file. After this we retrieve the values of $userid and $auth_key which will be required by SARK Auth for logged in users. For other cases it is null and passed to the class main_functions where it is accessed by access_api() function. Later we create

objects for database and main functions classes. And the later major task we do is to create an object of the smarty class and initialize it. We can initialize other classes here as well which can be required over the application. So this is the basic config of the application code. Now we shall look into next major file which is the site_functions.php and authorize.php file.

In includes directory we create an ajax directory which can contain all the ajax codes files that is required in the application. Lets look into brief what is Ajax and why we use it. Ajax is known for Asynchronous Javascript and Xml. This is required when we need to process any data without reloading the page. So a javascript handler calls a PHP page and waits for its response to update the contents. We know javascript is asynchronous and so the next javascript lines will keep on executing while the result in received from the PHP page and the page can be updated from its success block. Lets look into a simple block of ajax code below which are generally part of javascript files and forms a very important part of any PHP application.

```
function submitForm(formname, submit_url, divid='')

{

   $('form[name='+formname+']').on('submit', function (e) {

      e.preventDefault();

      var formData = new FormData(this);

      $.ajax({

         type: 'post',

         url: submit_url,

         data: formData,

         success: function (res) {

            result = JSON.parse(res);

            if (divid !== '')

            {

               // Update the page or show success / error message
```

```
            }
            if (result['url'] !== undefined)
            {
                // redirect the page
            }
        },
        error: function (error) {
            result = error;
        },
        cache: false,
        contentType: false,
        processData: false
    });
  });
}
```

The above function is a form submit code which submits and process form contents with a ajax call and updates or redirects the PHP page as required. This is generally written in js_functions.js file in resources > support_files > js directory. The location we shall check later but for now lets look how it works. When a form is submitted this function is called and the parameters formname, submit_url and div_id are passed. div_id is the html id of the div element where the response message to be displayed. Formname is the html name of the form that needs to be submitted and lastly the submit_url is the url of the PHP page that needs to process the contents. The data of the form is captured using FormData and this is followed by processing the data.

In ajax we pass the submit_url as endpoint, form data as data and process the data in PHP page and receive the result in success block. In case there is some error there is an error block to process the same. Once the result is received we can update the page or

redirect as required as we can see in the success block. Generally the best practices is to send the response from PHP page in JSON format as for javascript it is easy to understand and further proceed. JSON is infact an universal data transfer entity which is very useful for modern day applications. The PHP page that would process the ajax request should be placed in includes > ajax directory.

The includes directory can also contain essentials directory which contains the site_functions.php file and the includes directory also contains the authorize.php file. The site_functions.php file is a class based file which can contain multiple classes that processes anything that needs to be in functions in the classes. This site_functions.php file must be included in every file where the task needs to be done like top level PHP files which are directly accessed by the browser URL. So this the job of this file to have functions to process various tasks needed by the application side. While we need something to be performed from the API side we can call the API endpoint using access_api function in the functions.php file. This can be from ajax PHP file or the site_fucntions.php file. Apart from this the next file that can be contained in the includes directory is the authorize.php file. This file performs the task of checking and prohibiting URLs or pages that needs user login and sets the session data for those pages with $userid. If the user is not logged in it redirects the user to login page. Otherwise it retrieves the user data and make it so that only the user can view his respective data and not for any other users private data. Lets look into the code of this file and briefly study the usage of each lines of code.

```
if (!isset($access_key) or $access_key != 'the_sark_tech') {

    // redirect to 404

}

if (isset($_GET['referrer'])) {

    // redirect to referrer

}

if (isset($_COOKIE['username']) and isset($_COOKIE['password'])) {

    // get the encrypted username and password from cookie

    // decrypt the username and password and store in a variable

    if (!empty($username) and !empty($password)) {
```

```
            // check if the username and password are valid from API
            if ($status) {          // if valid
                    // get the userid and auth key and set it to session
            }
        }
}
$REQUEST_URI = explode('?', $_SERVER['REQUEST_URI'], 2)[0];
// get the current page uri from request uri
if (($current_page_url == <login>) or ($current_page_url == <forgot_pwd>)) {
    if (isset($_SESSION['userid']) and isset($_SESSION['auth_key'])) {
         if (!isset($referrer) or $referrer == '') {
            // redirect to home page
        } else {
            // redirect to referrer
        }
        exit;
    }
} else {
    if (!isset($_SESSION['userid']) or !isset($_SESSION['auth_key'])) {
        // redirect to login
    }
}
```

This whole code performs the task of authorizing the user from pages he can access. For logged in users it makes the user stay on the page with his data while if he is on login etc page he is sent to home page. While if the user is not logged in and he is in some protected page he is sent to login page. It also validates the users cookies in the whole process. Lets look into some important lines of code in the authorize.php file.

The first line is checking the access_key variable and preventing direct access to the file through URL. It is followed by checking the referrer from URL parameter and retrieving it in a variable in case it is set as this code is included in every page in the application server. Following this the cookies are checked and in case they are set they are retrieved, decrypted and verified from the API server to log in the user. In case user is already logged in the later part checks the same and redirects the user to login or home page or allow access to the pages as per each case. By studying the whole code it can be easily understood. So by using this authorize.php page the user authentication and authorization for whole of the application server can easily be achieved. By this we conclude the contents of the includes directory. Now lets move on to the next pages and directories in our application directory.

Next directory in the application is the libraries directory. This can contain the Smarty library and any other library that might be required in the application like PHPMailer etc. These libraries can be placed in the libraries directory. Next in the list is resources directory.

Resources directory contains three sub directories namely default, logo and support_files. The default can contain any default images, files for the application that can be statically available for the application. This includes default user image or any other thing. Next in the list is logo which contains the application logo in various dimensions and formats along with the favicon. This would be needed to be displayed in various pages of the application. Next in the list is support_files directory which contains the custom javascript and css files in the application. These would be in the js and css folder in this respective support_files directory.

Firstly lets look how we can or more properly I follow for the js directory. The js directory can consist of js_functions.js and ready.js file along with any other javascript file that would be needed in almost all the application pages. Otherwise javascript files for specific pages we can place them in a pages directory in the js directory. The js_functions.js file can contain the submit_form() method along with any other methods that would be required throughout the application. The ready.js function majorly contains the JQuery $(function){} to execute the javascript codes only when the document is ready. Besides it also can contain other functions. Now the css directory in the support_files directory can primarily contain two stylesheets namely custom_style.css and custom_style_mobile.css. The later can be specific for mobile

screens in case any adjustments needs to be made. So this is the basic configuearion for the support_files directory and the specific javascript and css files can be included in our page as per needs. Now lets look into the next directory which is the themes directory.

The themes directory generally contains the smarty template directory and email_templates directory. In email_templates directory we can majorly contain the email template files that can be used as email body while sending mails from the application. We can choose any suitable extension for the files viz .etpl etc. The smarty template directory would be root for the smarty templates which we shall create and call in the smarty object in our PHP code. This template directory can contain the compiled directory which would contain the compiled smarty templates into PHP files. So in the template directory we can create the smarty template files. To note we can create multiple template directories with different names when we would need to rotate the UI design from time to time.

Now the last directory is the uploads directory which can contain user uploaded files uploaded specifically by every user. A custom .htaccess file can be used in this directory to restrict and protected the user files from unauthorized usage. So this is the whole design of the application directory. The root of the application directory can contain the main PHP files which needs to be directly accessed from URL and would be starting point of any of the scripts. So this is how we can create a good system design for the application which is easy to maintain, secure and readable. This concludes the system design part of the application. We have also looked into the working of the SARK Auth authentication library. Now we shall move and look into the API system design part.

The API system is another vital part of our project as I mandatorily use APIs in any of the dynamic applications I create. Lets discuss the advantages of using APIs in projects. It provides flexibility, reuse of components and ability to create multiple endpoints as per requirements. But while using API architecture it is very important to keep in view the point of security and while using SARK Auth in proper way this can be implemented easily as we looked earlier. So by keeping all this in mind lets look into the structure of the API directory below.

```
api_dir
    config
        config.php
        functions.php
        values.xml
```

includes

essentials

site_functions.php

authenticate.php

user_authenticate.php

libraries

resources

libraries

sark_auth

<!—SARK Auth Library Files --!>

uploads

<!—User Specifc Uploads Made Through Application--!>

.htaccess

<!—PHP Files Accessible Through URL--!>

So this above structure is the structure we can implement as a part of system design for the API directory. Here also lets discuss all the files and folders and why they are used starting the config directory.

In the config directory there are three files values.xml, functions.php and config.php. These files are similar as for the application directory. But here in functions.php we would not require the access_api() function so we can skip that and rather use any other function that would be necessary here. Besides in config.php the smarty class is not required hence is not available while the rest of the constituents are quite similar as in the application directory like setting the defaults, base_dir etc. Hence lets move on and lets look into the includes directory which is somewhat different here as the authentication mechanism for the API is different.

The includes directory has the files authenticate.php for API authorization while user_authenticate.php for user based authentication while it also has a essentials

directory which would contain site_functions.php file containing the various operative function based classes for the API and the application. Lets firstly look into the authenticate.php code and how it can be used for API authentication. To be noted it is used concurrently with SARK Auth library.

The codes we have seen in earlier section of SARK Auth. We create a object of Authenticate class and verify the values of credentials and secrets with the ones in the library in an encrypted form. If any of them fails we stop the execution with a 401 unauthorized error otherwise we make the code reach the next section and process the request. Similarly for user_authenticate.php we authenticate the users for specific access. For this some secret is saved in database for the user id. We extract that secret, create an object of User_Specific_Autheticate class and verify these credentials for the user id. If verification fails we raise an 401 unauthorized error. Otherwise the request is processed. We need to keep in view that we use the user_autheticate.php only in those files where logged in user is trying to access some data. For other case authenticate.php is sufficient. So implementing these two authentication mechanisms using SARK Auth we can authenticate the whole API for authentication and authorization.

Lastly site_functions can contain different classes with functions that would process the request and make any changes in data in tables or generate a response value with some logic. This class based approach makes the code cleaner, extendable and secure. All these function can be called from top level PHP files which were directly accessible in the API server as per the URLs mentioned in .htaccess file. So using these we can implement the major functionalities of the API server that are contained in the includes directory.

Now lets move on to the resources directory which contains the SARK Auth library itself inside a sub directory called libraries. This makes the base of the authentication and authorization functionalities. The other contents of the API directory is the uploads folder which contains the user specific files and is occasionally used, the .htaccess files as explained for the application directory for protecting the server and routing URLs and root level files which would be the ones that can be directly accessed using URLs. So all these together can form the API directory and combining the application and API together we can create a complete app that can show and process data, request and display the response according to the requirements. This two can be hosted on separate domains and subdomains and form a complete ecosystem for the application.

Now lets move an and look into the admin directory structure which is often useful for maintaining the application by an admin UI rather directly accessing the database and code files. After that we shall look into some of the practices needed to be implemented while creating this complete architecture or system design. And lastly we shall look into basic logic of an URL shortening app which can be useful for creating a

complete PHP application by your side combining all of these. So lets look into the admin directory system design firstly which generally is very much similar to the application directory except the fact that it does not implement an API and most database operations are performed from the files in admin directory only. So lets look into that now.

The system design of the admin directory is similar to the application directory. It would contain the same pattern and compulsory files. But the site_functions.php file within the includes > essentials directory would contain direct database operations if needed or processing the inputs as we lack using the API design here. The rest of whole of the structure remains the same for application directory. We can create admin panels similarly to keep track of the application portal and also to manage it. By keeping this in contention lets move on to the next section where we shall be creating a simple page using application and API to look how we can use this design for creating applications. We shall look into the best practices by taking example of a URL shortening app.

So firstly we shall start from the .htaccess file of the application directory and use the below code in it.

```
RewriteRule ^shorten_url/?$ shorten_url.php [L,QSA]
```

We shall create a file shorten_url.php in the root of application directory. Then we can write the following code for processing the request.

```
include_once('config/config.php');

include_once($base_dir.'includes/essentials/site_functions.php');

include_once($base_dir.'includes/authorize.php');

$url_to_shorten = $_REQUEST[‘url_to_shorten’];

$shortened_url = $App -> shorten_url($url_to_shorten);

$smarty->assign(‘shortened_url’, $shortened_url);

$smarty->display('includes/header.tpl');

$smarty->display('url_shortener.tpl');

$smarty->display('includes/footer.tpl');
```

So this above code will pick the url to shorten from REQUEST and send it process in the function shorten_url in the class App in site_functions.php. In the shorten_url function we can call the API where url will be shortened and get back the response. This is just for demonstration while such simple logic we can write in the application code itself. So in site_functions.php we can write below code.

```
if (!isset($access_key) or $access_key != '<access_key>') {
   header('Location: not_found');
}
class App {
    public function __construct($upload_dir, $base_url, $api_url, $main_func)
   {
      $this -> main_func = $main_func;
      $this -> base_url = $base_url;
      $this -> api_url = $api_url;
      $this -> upload_dir = $upload_dir;
   }
   public function shorten_url($slug)
   {
      $url = $this -> api_url.'shorten_url';
      $data['slug'] = $slug;
      $shortened_url = $this -> main_func -> access_api($url, $data);
      return $shortened_url;
   }
}
```

Now we are sending the url to shorten to the API layer where it would be processed in a PHP file which also can be routed for a URL using .htaccess. The URI should be shorten_url as we mentioned in the function for variable $url. In the shorten_url.php file in API directory we can write the below code.

```
require_once('config/config.php');

require_once($base_dir.'includes/authenticate.php');

require_once($base_dir.'includes/user_authenticate.php');

require_once($base_dir.'includes/essentials/site_functions.php');

$response_data = array(

    'status' => 0,

    'message' => '<error_message>'

);

if (isset($_POST['data']))

{

   $postdata = $_POST['data'];

   $slug = $postdata['slug'];

    if (!empty($slug))

      {

         $slug_data = $portal -> shorten_url($slug);

         if ($status)

         {

            $response_data = array(

               'status' => 1,

               'message' => '<message>',
```

```
                'data' => $slug_data
            );
        }
        else
        {
            $response_data = array(
                'status' => 0,
                'message' => '<error_message>'
            );
        }
    }
}
echo json_encode($response_data);
```

This above code will process the slug in a function shorten_url in the class Portal in site_functions.php file in the API directory. We would create the class Portal and process the slug. The response data is returned in JSON format. I am not mentioning the site_functions.php file here for the class Portal. It can be easily created by following the specifications as I mentioned earlier. Similarly for request that can be processed from data from database we need to use the database_functions to alter the data and return the response to the application server. The processes could be made within site_functions.php file as demonstrated here. Now in the application directory the response would be received in the variable $shortened_url in shorten_url.php in application directory. The $shortened_url variable is passed to the template file where it can be displayed by proper Smarty syntax to the front end.

So this is a brief demonstration of how we can use the system design to create pages and full applications of our choice following the pattern as here. Similar is for Ajax requests which can easily done using submit_form function in js_functions.js file. The function can be called in ready.js file with proper parameters by mentioning the Ajax URL. In the Ajax URL the request is similarly send to API and a response is received

which is send back to frontend using the submit_form function. This is similar to the function in site_functions.php file. Only difference would data would not be loaded while page load rather will updated even without the need of reloading the page. By following these principles or adding any new that fits the design we can create any complex business application using the system design. So this concludes all the sections of the chapter on system design. Now we shall in brief look into some of the best practices before moving on to the next chapter where debugging PHP, Smarty, MySQL and JavaScript code would be discussed.

So some of the best practices would be to fix any logical error or any possible runtime error by logic itself. It includes multiple checking of the variables if they are null or have value when required. Proper error messages should be sent back to the UI for the user to better understand if there is any error. We shall look into most of the best practices or approaches later. So this finishes our discussion into system design and I believe by following this layered approach any complex business level application can be created bug free for the best user experience. So lets move on to the next chapter now which is we shall look into how in best way we can debug our applications.

DEBUGGING

So lets look into our new chapter “Debugging”. Here we shall look into the best approaches which we can take to debug our codes and make the applications flawless. So in this chapter we shall look into the debugging for the various categories and how to effectively debug them to create bug free applications which are providing good user experience. While debugging can be of three types – bug fixes, features validations and user experience check. We shall look into the good approaches to create such applications that comply with all three of these. So lets start our discussion.

So firstly lets start with bug fixes and the best approaches for the same. Bug fixes can be considered for these specifications which can be differentiated. In an application this can be for these specifications separately – PHP, MySQL, JavaScript, CSS and API responses. This would be checks for any bugs or run time errors. This can be implemented by code fixes which we shall see what can be the best practices. After this section we shall look into functionality checks and user experience checks which includes working of the code in various screens. So lets continue with the bug fixes.

So lets start with on debugging run time or even compile time errors of PHP. If there is any compile time error then the page itself will not load and with errors reporting set to enabled these errors with specific message and line no is displayed. Errors in PHP are of three types namely – errors, warnings and notices. Primarily errors and warnings must be fixed while notices are optional as code execution is not stopped for notices. To enable displaying errors and error reporting we can write the below code at top of the script.

```
error_reporting(E_ALL);              // sets to display all errors

ini_set('display_errors', '1');      // forcefully sets error display by overriding php.ini settings
```

So by writing any of these two lines we can set error reporting to on in PHP so that the specific errors can be displayed on screen. While error_reporting function will be effective if display errors is on in php.ini file. Again with ini_set function we can override the php.ini settings in our php code for the specific script. Now lets look into the approaches that can be effective in displaying and fixing the errors.

In case there is a specific error message in the browser we can quickly check on to the line number mentioned. Then we can try to fix the error which can be due to some unhandled value like null for the case. We might need to check if the value is correct or the value at the line number might be overridden by any part of code. With that we can check and process the code for null values also which must not throw an error. For checking wrong values and the portion code we can segment the code using exit functions and check from exactly which part the value is set as wrong. This can be done

by checking the value at different lines of code using following exit function. So this can get us to the exact line where the value becomes wrong so that can fix it. This is called the breakpoint approach. In case null value is the correct value we need to use proper checks to accept and process the value. So these are some of the scenarios which can be useful which debugging our PHP code. Some other scenarios that often comes are like maximum execution time exceeded. This type of errors are fixable by increasing the default values of the variables like max_exec_time in php.ini. This should be accompanied by restarting the Apache for the changes to take effect. Now we shall look into some other probable scenarios.

One of the approaches many developers employ is using the try-catch block. But I always suggest to use try-catch block only if there is a possibility of runtime errors such as connection lost etc which are unavoidable. In all other cases I suggest to employ sufficient logic to handle any errors that might arise. In fact this approach keeps the functionality not break in specific conditions and makes the user understand what actually is going in the application. Another suggestion I would suggest to use most of the PHP inbuilt functions in code which would always make the code more concise and effective. There must be proper validation for each and every possible case in the code to properly handle the possible errors. For API based systems this should be employed in both application and API as they should be thought as separate systems each should be wholly capable of handling any request itself properly.

Sometimes we may be getting errors for specific input values only and to debug those cases among large set of data we may be implementing conditional breakpoints like using the exit function for specific inputs. Besides these we can create log files for production environments where display errors might be set to off to properly log in case any error is still there but skipped our testing. By default the servers contains PHP and Apache logs which can also be beneficial for the purpose. Besides this for new programmers can be using VS Code for checking in case there is any syntax errors which itself get caught in the IDE while writing code.

While testing the code we must try to check with all possible cases leaving not a single behind which needs thorough planning and investigation. So while developing apps we must focus on modular way and while debugging any module our focus must be on that module only. Now taking the case of the system design I explained in previous chapter sometimes the API has error but it does not return. So while trying to debug that from application we can directly print the response as that can show the exact error on API side to fix it. Similar methods can be used for any API based systems. Besides that while debugging a modular approach is always beneficial because we can check for the errors in a series of steps to find actually where the error is. In addition to this there are various tooling and debugging tools available as extensions in VS Code like XDebug PHP Extension etc which can be installed.

In PHP generally two types of errors needs to be fixed – errors and warnings. Errors needs to be fixed because they will stop the execution and cause unnecessary outputs. While warnings are something that though not now but in sooner versions or can cause errors in output and hence we can not rely on the presently written code. So these two types of messages are compulsory to be fixed. Now lets move on to elaborate on how to fix functionality errors in the application if there is any.

For functionality errors there is a new approach we need to use. Firstly according to the requirements thorough testing with proper data sets needs to be checked that the feature is working as expected. This needs to be checked with various data sets that even if no error may be displayed but the output must be same as expected one. We should also keep note that one of the ways to test is to do unit testing by writing automated test cases. But we shall focus on manual testing by developers and check the approaches that can be taken for making the application totally work as expected. Firstly we should create a data set that covers all the possible scenarios and check them for correct output. This properly validates the application will run beg free in production. Proper test cases in cases of CI/CD pipelines can automate the process and deployment can become faster. Also to note for testing only APIs as per use case we can use proper API testing tools too like Postman to debug APIs. So these are some approaches for functionality testing. But again I should focus on properly test the application by each module end to end before deploying to production. Now lets move and check how we can debug the dynamically created MySQL queries in our application.

For checking that any MySQL query is running without errors or returning correct data firstly we need to print out the output after executing the query. In case no data is returned we can print the SQL query itself. In both cases in case of any issue the best approach is to copy the SQL query and execute it directly in MySQL to check the error and fix it accordingly. Only if the connection is correctly creating then this step needs to be done otherwise we need to check why connection is not being created and fix it accordingly. For proper data we can try different clauses or conditions as required and try to solve it. If there is any error in query then the error will be displayed in MySQL and we can fix according to the error. Some of the best approaches for MySQL queries is using limits on data based on pagination. This will make data retrieved quickly and prevents connection lost or timed out errors. This we shall look in upcoming chapters. So now lets move on to our next section of debugging JavaScript code.

The most important factor to note is that in JavaScript if any error shows up then further JavaScript code will not executed but it will not stop the PHP code and no error is shown on the page. Rather to view the error we would need to check the browser console. Hence every time we test a page then we must check the console for errors along with the page. This will ensure that any JavaScript error too would not be existing in the code. Lets now look further debugging techniques for JavaScript codes.

Firstly we need to check the browser console and look up for any errors. We should follow the error message and follow the line no and file specified and fix the JavaScript code. After fixing we should check the console again along with the functionality that the module is behaving as expected and proper results are being displayed on the page. For proper debugging we can use console.log statement at various places to print out the variables and see which value is breaking the code. Besides we might need to check if cookies are set correctly by checking the cookies in browser. These are some of the must know rules for debugging JavaScript. Besides we can check the network tab for responses for Ajax calls from PHP scripts. This can display if there is any error from Ajax script and can be fixed in the PHP script itself. So these are some of the common techniques that needs to checked while debugging JavaScript. We haven't covered JavaScript and CSS in this book and for complete reference to these languages you can check any books specialized to these. Now lets move on and check how to debug CSS code briefly.

CSS or Cascading Style Sheets are codes that makes the front end look good by adding a proper presentation to the html elements. For reference to CSS I would suggest try out any book with complete reference to it. Here we shall look on briefly how to debug the same. The foremost requirement is to check the user interface is looking as expected in front end with proper design but not to forget it to check for different screen sizes and see if the interface is still as expected for the screen size. So I suggest to create a custom CSS file for the whole application to modify over the theme files if any required and another custom CSS for mobile that can override the custom CSS and work for smaller screen sizes which makes it easier to code as different CSS may be required for different screen sizes and that can be accomplished using the technique. To fix the CSS you need to study CSS codes and why they are used and for that I would suggest to pick a suitable book. So this concludes the debugging for JavaScript and CSS files. Now lets look into how we can be validating the functionality and user experience for the application as a whole.

For functionality the best approach is modular end to end testing. These should be accompanied by almost all possible test cases and if it works well then can be moved on to next module. In case of error these should be continued with further debugging as explained earlier. The end to end testing should be accompanied with PHP, JavaScript and also CSS code to check with the functionality and display for if anything needs to be fixed. In this manner picking up each module the whole of the application needs to be tested to see if the application is without any errors and can be deployed to production. But before deploying another factor to be kept in check how would be the user experience.

User experience checks should be firstly made during the initial stages before starting of the development and the whole of the application should be developed based on that

accordingly. But still once whole of the application is developed and before deployment the user experience must be rechecked if that is actually user friendly and without any more modifications can be provided to end user. That should be properly validated that even if the design and development has been as per specifications but still the user would be finding it easy and convenient to use. Upon getting confirmation on the same upon carefully testing that the user experience is good enough to release then only the deployment should be proceeded with. So upon properly validating all these mentioned checks the application can be proceeded to be deployed over production.

So all these complies are required for proper testing and debugging of the application before release. In upcoming chapters we shall study on best practices to write safe and bug free code. While in next chapter lets look into Git musts for collaborative environments. So this concludes our discussion on debugging and lets move on to the next chapter.

HOW TO USE GIT

So lets look into our new chapter “How to use Git?” for collaborative environments. So what actually Git is? We shall look into that in this chapter along with what are various environments for Git, how to setup for Developers, various important Git commands, what are CI/CD pipelines eg GitHub Actions and what additional features available in Git. These will provide us a good idea about Git and how we can utilize it for our needs. Git is basically a tool that is used to help people or teams work on the same project in a collaborative manner. Different people work on different modules of the project and they update the code in the repository simultaneously without anyone affecting any others code. This is the biggest practical advantage of using Git. So lets now first look what is Git?

Git is a version control system in which we can manage different versions of our application as separate instances along with different people in team collaborate to update the project simultaneously. The git repositories can be accessed by git commands and different developers can update code which is also kept tracked. This creates concurrency, transparency along with speeding up the process. Git is actually also a centralised repository. So this is basically what Git is. Now lets look how actually Git works as a whole?

Git is actually a centralized repository where different versions of code is managed. The Git repo has a url that is common to the repo and different people across different locations can access the repo using that url. The Git url looks something like as specified below.

https://<git_host>/<owner>/<repository>.git

This is the format of Git url. The host is the host like GitHub, GitLab etc. The next part is the slug for repository owner while the last part is the repository name with .git extension. This is how the Git url is configured. Now on this url different versions of the code is maintained which are called branches. When we fetch the code from git by default the master branch is fetched if we do not specify any particular branch which we shall check shortly. So lets look into the complete flow of Git now.

First a repo is created in Git say GitHub. A production branch is created from where production code is deployed. Other branches are staging in which code is tested and multiple developer specific branches. Now each developer works on his branch and updates the code in his branch in the repository. Now after verifying that code is merged with the staging branch. Merge and fetch are very important functions in maintaining a Git repo. So once the code is merged in staging similar is done with multiple developer branches my keeping in view no conflicts stays. Once this is done the code is tested from staging which needs to be deployed in a test environment. Once all looks good this branch is merged with production branch which is then deployed to production. If

testing in staging has errors then the developer takes a pull (as in git) which is actually a fetch operation from the staging environment to his development branch and reworks and again updates his specific branch in the repository. The whole process is repeated. This is how the whole process of using a git repository works. To note few of this steps are done manually and it is without CI/CD pipelines. In CI/CD pipelines this whole process of merge and test is performed automated using test cases and is available in features like Git Actions which also we shall look later. Now we shall look how the process works in more detail but before that lets look into the various Git environments available.

For the environments there are various Git providers who can be used for commercial or enterprise level. For example GitHub, GitLab etc. GitHub is one of the oldest and most important among them. We shall reference GitHub in most of our discussions. These are centralized architecture where different enterprises can host their own repository for access to their teams. By keeping this in view lets discuss what type of setup is required for developers to use Git.

Firstly the enterprise creates their Git repository and provides admin access to their DevOps team. The admin has all the access to provide various access to other members like read, write, triage and maintain with respect to GitHub. Now a user according to his requirements are provided specific accesses so that he can perform his tasks. Now a developer requites a read and write accesses for fetching and updating his code on the repository. By read access they can fetch the code to his local environment which is known by clone / pull operations and work on it. While by write access they can push or update the code in the repository and create standard pull request which is followed by merging the developer branch to the staging branch. These whole processes are performed by a series of commands known as git commands which we shall look shortly. Generally the developer works in his local or the enterprise environment where the code is fetched and from where updated each time. To note whenever there is any update in staging or base repository the code should be fetched by local code using pull commands. Also while merging two branches there can be conflicts like different code on same line which is not committed means finalized. In that case we need to keep one of the code in the line. This is called resolving conflicts. This can happen only if last merge is not committed. Commit is a command in Git which finalizes that code. Now lets look into the major Git commands and why they are used before summarizing the working of Git keeping in view of the commands.

So lets look below into the important git commands firstly.

- git init
- git clone <url>
- git status

- git add <file>
- git add .
- git commit -m <message>
- git branch
- git branch <name>
- git checkout -b <branch>
- git checkout <branch>
- git fetch
- git merge <branch>
- git pull
- git pull origin <branch>
- git push
- git push origin <branch>
- git log
- git stash
- git stash pop
- git remote -v

So these are the major git commands that are useful. Besides these there are also a lot bunch of git commands that can be looked into from the internet. We shall go through each of these commands at this moment and look into why they are useful.

The first command with git we can use is “git init” command. This is used to initialize an empty git repository in local environment. This is generally occasionally required and not for already existing repositories.

Next “git clone <remote_url>” is one of the very important commands that is used to clone or copy the remote repository to the local work environment. This would clone all the branches of the repository of remote url on the local environment. We can create a new branch afterwards or work on an already existing branch by switching to it. We can clone only particular branches as well as required using --single-branch option. There are a few variations as well which can be checked out from the internet. The remote_url to clone can be found out from the repository on GitHub, GitLab etc.

Next command “git status” command is used for getting the present status of the git repo for the specific branch. It involves committed and uncommitted files, added and not added files, tracked and untracked files etc. This is used very often by developers to check what is the status of their repository branch and the files they have worked on and yet not committed to be pushed. Sometimes it is required to check other things as well.

Next command is “git add <file>”. When we would need to add a single file to commit queue then we can use the file path. While if we need to add all the files that are

modified and uncommitted we can use git add . which adds all the files to queue. These files that are added can be added to commit afterwards.

Next command “git commit -m <message>” is used to commit the code for pushing it to the remote repository to any or generally the same branch. This must be with a commit message which specifies the motto of the work. Generally for any specific task one commit is necessary and in the message we can mention the details of the task for that commit. Once committed this needs to be pushed to the remote repository and branch.

Next command “git branch” provides the current working branch on which the developer is working. It only lists the branch name. Though when used with a branch name as “git branch <branch>” a new branch is created with the branch name specified. Though still the working branch is not switched to the newly created branch rather old one and needs to be switch manually.

Next command “git checkout” is used for switch to a new branch with the changes that were made in present branch. This helps in keeping the changes in the new branch code. To create a new branch while checkout we need to provide the -b parameter. This will create a new branch from the current branch and switch to the new branch along with the changes.

Next command “git fetch” download the latest commits, changes etc from the remote repository without merging them to local instance. This needs to be done manually.

Next command “git merge <branch>” is used to merge an existing feature branch into the present branch. Generally there is a base branch in which we need to merge other branches. While merging there can be conflicts in case there is some uncommitted changes in current branch and new branch too has on same line of code. In such cases we need to manually check them keep any of them and commit the code back. After merge the code at current branch will have all changes from feature branch which is committed. Generally in remote repository any developer’s branch is merged to the base branch in a similar fashion.

Next command “git pull” is a very important command and often used to update local repository code. The default command without any parameters will bring all the changes from same branch in remote repository and merge it in the local. When used with parameters as “git pull origin <branch>” it pulls the code from mentioned branch in remote repository and merges it to the local current branch code. This is often used to keep track of the changes and update the local code with remote repository code in case it has undergone any changes.

Next command “git push” is another important command which is used very often. This is used to update the code in remote repository with the local code of developer for the same branch. In case it is used as “git push remote <branch>” the local code in current branch is update to remote repository code of the branch which is mentioned in the command. This automatically merges the code as well. This is also used once development is finished at user end and code needs to be deployed.

Next command “git log” retrieves the commits history for any specific branch. It helps in identifying in case any rollback is required at a later stage.

Next command “git stash” is used to temporary remove uncommitted changes and saves them in a local file which later can be retrieved. This is required when we want to test with new code and if new code fails we can revert back to the old stashed changes. This command is accompanied with “git stash pop” which retrieves the stashed uncommitted changes back into working directory so that work can be carried from there itself.

Next command “git remote -v” is used to recover all configured remote repositories along with their tracking urls. This is required in case we need to quick get the repository url for the local repository.

So these are most of the very useful commands that are required very often while working on git. Generally merge access is not there to developers in remote repository. They need to create a “Pull Request” of their branch to the base branch from the GitHub. Once done the developer’s branch can be reviewed and merged to the base branch in remote repository. So this is how a developer can use the commands to collaborate in a team to work on a project. Now we shall look at the common steps in brief of using Git followed by automating the Git workflow using CI/CD pipelines.

Suppose we need to work on a new feature on an project. The project is hosted on a repository with some url. So the developer comes in and how he would make the changes and send it for deployment? Lets look.

Firstly the developer will clone the repository to his local instance. This would be followed by they checkout to a new branch say feature-branch from base branch. Once done they would work on the feature and make necessary changes in code in his local instance. Then they would test and then add and commit the files in his local instance. In case they got informed that remote base branch is modified they would need to take a pull from base branch of remote. Afterwards they will push their branch to remote repository and need to create the branch there. Thet would need to pass the parameter -u in the push command along with the branch name. Once the branch with the committed changes is pushed to the remote repository they need to raise a PR (Pull Request) for

their branch with the base branch. Now the reviewer will review his code and if its fine merge it with base branch which would be staging. The application is tested on staging and if its fine it is merged with production branch from where the code is deployed. So this is the complete workflow of the Git for developers. Now lets move on and look into how we can automate this workflow using CI/CD pipelines like GitHub Actions.

We shall only look into the basic idea of GitHub Actions here. Basically in CI/CD pipelines the deployment is automated by automating the merging and testing part. There are various tools for different providers. Similarly GitHub has GitHub Actions for CI/CD. CI/CD stands for continuous integration and continuous development actually. This works by separate triggers for code push, pull requests etc like when they are created a series of automatic processes starts. This includes code being merged followed by testing the code using already created unit test scripts on passing which the code gets merged and further deployed to production. This reduces deployment time subsequently and make the work of DevOps easier once the whole process is set up. So this is all how CI/CD pipelines like GitHub Actions works. The core components of GitHub Actions includes Workflows, Events, Jobs, Steps and Runners. For detailed understanding of all these I would suggest going through any book of complete reference on CI/CD pipelines or search over the internet.

So by discussing all these we have covered specifically what is Git, how it works, various commands, deployment cycle and CI/CD pipelines. So this completes our discussion on Git. In next chapter we shall look over the best approaches that would be essential in writing good standard code. This would be very beneficial as maintaining specific rules and standards we can maintain good readable, extendible code which also would be bug free and provide a good experience to end users.

BEST APPROACHES

So lets look into our new chapter “Best Approaches for Writing Effective Code”. So let’s begin. So when we say best approaches we generally mean that the code should be bug free, secure, easy to read and maintain. We shall look into the approaches that can make our code comply with these things. So lets point out what are the things that we need to take into contention while writing our code. The points I would like to highlight would be validation, error handling, try-catch usage, code indentation, object oriented, specified functions, security and maintaining structured approach. Lets look into each one by one.

So lets begin our discussion with how to properly validate the inputs. First thing we need to keep in mind that before processing any input we must properly validate it for empty etc checks. This is called validation. For API based structures the validation should be done in both application and API sides. We should handle application and API as separate entities and both should possess the capability to individually validate and return proper response. This is how we should properly validate every input before sending for processing.

Next thing we should employ is proper error handling mechanisms. This specifically means each peace of invalid input must be processed and proper error message should be displayed to the end user. Any input that may raise an run time error in the application must be handled before that and proper error message should be displayed to the end user for that. It makes the user experience good along with prevent the application to break at runtime. Again the API and application end both should have the same error handling mechanism. So error handling is another important part in developing complex real world applications.

The next thing we should note that many developers use try-catch block unnecessarily at every place of their code. But actually while it does not display errors in the user interface but can make improper outputs which is too should not be. So always to be noted that try-catch should be in code where there is a possibility of a runtime error that is beyond our control. Otherwise we must maintain proper checks and error handling mechanism for all other types of errors. The try-catch block usage should be in places like database connection lost, file server connection lost etc. This thing can be handled by try-catch block. But improper inputs or any other errors that can be handled by code must be done so using proper error handling mechanism.

Next thing we should keep is proper formatting and indentation of the code. This is because this makes modifying the code easier by making it easier to find error segments in code. A neat and clean code makes any update in the code or even reading the code easier. So this is always we should keep in our mind while writing any code.

Next we should always look to make any application object oriented. This is because in that case the code is far more easier to maintain. The code stays modular and also reusable. So any specific change we need we can make in that module only by not even thinking of any unnecessary issues in other modules due to that. So a object oriented approach makes the code reusable and make specific functions for specific tasks. This approach makes the code extensible, easier to find and fix bugs and also the code readable and reusable. Hence a object oriented approach makes the development much more easier for complex business level applications.

Next point to note is to use separate functions for separate tasks inside classes. This is the very low level unit for modularizing the code. This helps modification, bug finding or development of any module independent of the other hence extending the application or modification easier too. So again this approach also helps code readability easier and a very good point to note while building applications. Every module or task should be completed using separate functions dedicated to same.

Next very important point we should be figuring is the security of the application. We must maintain direct access to PHP files to not be by using specialized code like access keys as we checked earlier. Besides directory or unnecessary files should be protected using .htaccess. As in an application user data security is of prime importance so this is one thing that that should always be focussed on while creating applications. Except these security mechanisms any other security like protecting user uploads should also be protected by using specific .htaccess file dedicated for the same.

Next important point to note that we should be following a structure approach for the whole application. This can be implemented as the system design we have studied earlier. So following such proper structured approach we can easily build complex business level structured applications.

So by looking into all of the above mentioned ideas we can easily develop complex business level scalable applications. Because all these points are very important and we should not be compromising with any of these to properly build such applications. By keeping this in mind we can proceed to next chapter and check out the summary that is what all we have discussed throughout the book in brief. I believe so far what we have discussed is very much sufficient to guide you in building such business level applications. But I would always suggest you should always be trying hands on which will definitely improve your idea and experience in the development of such applications. On this note lets move on to the next chapter and check out the summary section.

SUMMARY

So lets look into final section of our book which is the summary section before completing our discussion. In this chapter we shall discuss what all we have studied and all of them as whole can benefit us in which manner for our motive that is developing complex business level scalable applications. So lets briefly go through firstly at what we studied.

Throughout the book we have respectively studied how to set up development environment, basics of PHP, Smarty and MySQL, system design, basics of Git and debugging and best approaches for developing our applications. So by combining all these along with a go through of JavaScript and CSS I believe we can have complete idea for the motive. While we can study coding for anywhere I believe system design, debugging and our approaches are major factors that decide how good of a developer we are. Besides our skills in Data Structures and Algorithms (DSA) are the backbones of any developer as they are needed in lower levels of the application. With enhancing these skills by proper hands on and logical thinking any very complex application or feature can be developed easily. With these and AI tools which are also on trend while I am writing this book coding is going to me at all new level while developing new applications. Even creating AI tools or AI applications is very easy by following the steps mentioned in this book.

While writing this book my motive was to make developers able to create business level applications that can solve real world problems. This comes from my expertise in which I also have had created many such complete applications solving real world problems and I thought to share my knowledge and experience to the readers so that they can be too be confident and able to create such applications. The world is changing every day and need of newer and newer applications are arising every day. Every thing is looking to get automated and software are only means to do that. In contrast to that web applications provides a centralized system of such.

With increasing popularity of NodeJS, Python, Java etc for developing applications PHP actually has never lost its foot or capability for serving the world wide web. In fact PHP has also evolved. But I believe the lack of pre available libraries and more of coding from scratch necessity has made PHP a little bit unpopular. But again In believe the best developers would like to take control and for that developing from scratch is an very important part. Hence for the best developers in the job using PHP is always a better option to use to develop complex real world applications that solves real world problems.

So in the chapter of basics of PHP, Smarty and MySQL we have covered most of the things we need to become a good developer. Along side this we have kept focus on system design, error handling, debugging and best approaches which can make us all in one developer who can independently work and create such applications that solves real

world problems. Even we have looked into some of the collaborative tools and automated collaborative tools which we also might require in some of the cases. By keeping all this in mind I believe we can be confident on work on real world applications and create something that truly matters.

With this note I am completing this book with an advice for trying hands on over all of these mentioned in the book. I believe any level of development you have done so far after reading this book you must be confident of creating such complex business level applications that solves real world applications. As real world problems are increasing so we need developers who can work and solve them with software. And that is also the motive and motivation behind this book. So I finish by discussion and be glad if you took the responsibility to work on such real world problems and provide solutions for all the people around. Congratulations for doing the same!

END

ABOUT THE AUTHOR –

Shouvik Sarkar is an Entrepreneur, Author and Programmer. He was born in Oct, 1990 at the city of Kolkata. He has owned a software firm earlier during 2019 – 20 named The SARK Technologies. He has written couples of books so far previous to this. Besides he has developed a lot of software and working on some major one's too.

This book is a completely technology and programming based book dependent on his experience and knowledge. He has been planning and looking for starting new companies in future related to technology sector to be based on his knowledge.

To contact him you can reach to him at his email mentioned below or you can contact him through his website.

Email : shouvik@shouviksarkar.net.in
Website : https://shouviksarkar.net.in

Shouvik Sarkar