Android app Full Stack Development Course

Use real app code example to teach java programming
Simple Login Register, PHP API and database

Java

Variables

Identifiers

Primitive Data Types

Casting

Method Parameter Class Object Constructor

if else switch While For Break Statement

MultiThread runnable

Inheritance - extend

Overriding

Abstraction

Encapsulation - Public Private Protected Package

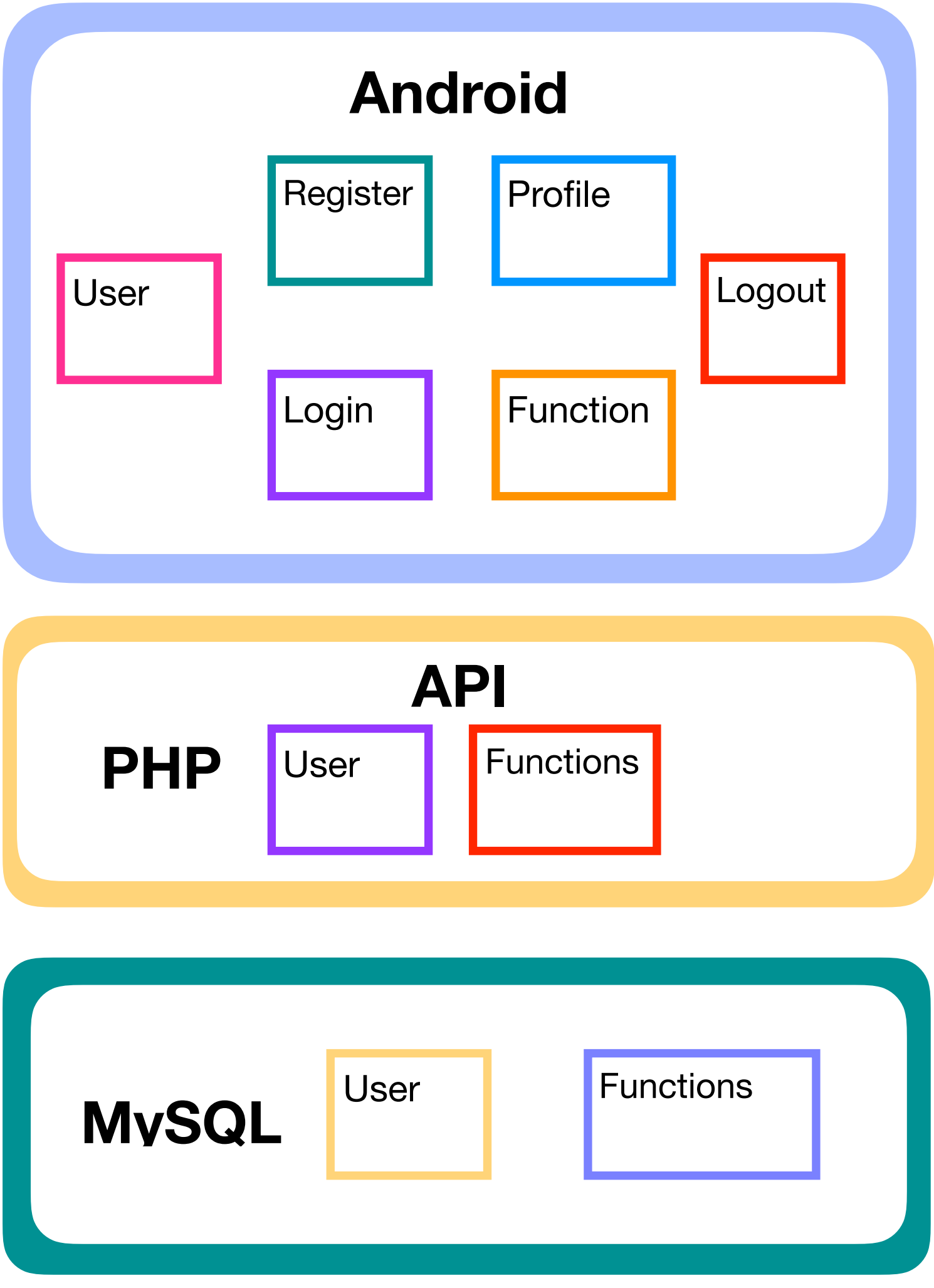
Polymorphism - Inheritance method have many form

Interface - implements

Package

Array Arraylist Vector Hashtable Enumeration Stack

Reserved Word


Android
Register
Profile
User
Logout
Login
Function
API
PHP
User
Functions
MySQL
User
Functions

What is Java?

```
public class MyClass {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Java is a popular programming language,

It is owned by Oracle, have Million Device Running

It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection
- And much, much more!

Get Started

This course can learn the basic Java language for android Development

Java - What is OOP?

OOP = **Object-Oriented Programming**.

object-oriented programming is about creating objects that contain both data and methods.

Object-oriented programming has several advantages

OOP is faster and easier

- OOP provides a clear structure for the programs
- OOP makes the code easier to maintain, modify and debug and Reuse
- OOP makes it possible to create full reusable applications with less code and shorter development time

Class Object Method

```
public class User {



    private int id, user_counter;
    private String username, email, gender;

    public User(int id, String username, String email, String gender, int
user_counter) {
        this.id = id;
        this.username = username;
        this.email = email;
        this.gender = gender;
        this.user_counter = user_counter;
    }

    public int getId() {
        return id;
    }

    public String getUsername() {
        return username;
    }

    public String getEmail() {
        return email;
    }

    public String getGender() {
        return gender;
    }

    public int getUserCounter(){
        return user_counter;
    }


    public void setUserCounter(int user_counter){

        this.user_counter= user_counter;
    }

    public void addUserCounter(){

        this.user_counter= this.user_counter + 1 ;
    }

}
```

MainActivity

```
package com.myclassstudio.album;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.content.Intent;
import android.view.View;
import android.text.TextUtils;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ProgressBar;
import android.widget.RadioButton;
import android.widget.RadioGroup;

public class MainActivity extends AppCompatActivity {
EditText editTextUsername, editTextEmail, editTextPassword;
RadioGroup radioGroupGender;
ProgressBar progressBar;

public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";


RequestQueue queue;
JsonObjectRequest request;
Map<String, String> map = new HashMap<String, String>();


private RequestQueue mRequestQueue;
private StringRequest mStringRequest;



@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    progressBar = (ProgressBar) findViewById(R.id.progressBar);

    //if the user is already logged in we will directly start the profile
activity
    if (SharedPrefManager.getInstance(this).isLoggedIn()) {
        finish();
        startActivity(new Intent(this, WordpressPostActivity.class));
        return;
    }

    editTextUsername = (EditText) findViewById(R.id.editTextUsername);
    editTextEmail = (EditText) findViewById(R.id.editTextEmail);
    editTextPassword = (EditText) findViewById(R.id.editTextPassword);
    radioGroupGender = (RadioGroup) findViewById(R.id.radioGender);


    findViewById(R.id.buttonRegister).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //if user pressed on button register
            //here we will register the user to server
            registerUser();
        }
    });
```

```
    findViewById(R.id.textViewLogin).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View view) {
            //if user pressed on login
            //we will open the login screen
            finish();
            startActivity(new Intent(MainActivity.this, LoginActivity.class));
        }
    });
}



private void registerUser() {
    final String username = editTextUsername.getText().toString().trim();
    final String email = editTextEmail.getText().toString().trim();
    final String password = editTextPassword.getText().toString().trim();

    final String gender = ((RadioButton)
findViewById(radioGroupGender.getCheckedRadioButtonId())).getText().toString();

    //first we will do the validations

    if (TextUtils.isEmpty(username)) {
        editTextUsername.setError("Please enter username");
        editTextUsername.requestFocus();
        return;
    }

    if (TextUtils.isEmpty(email)) {
        editTextEmail.setError("Please enter your email");
        editTextEmail.requestFocus();
        return;
    }

    if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        editTextEmail.setError("Enter a valid email");
        editTextEmail.requestFocus();
        return;
    }

    if (TextUtils.isEmpty(password)) {
        editTextPassword.setError("Enter a password");
        editTextPassword.requestFocus();
        return;
    }



  StringRequest stringRequest = new StringRequest(Request.Method.POST,
URLs.URL_REGISTER,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    progressBar.setVisibility(View.GONE);

                    try {
                        //converting response to json object
                        JSONObject obj = new JSONObject(response);

                        //if no error in response
                        if (!obj.getBoolean("error")) {
                            Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();
```

```
                            //getting the user from the response
                            JSONObject userJson = obj.getJSONObject("user");

                            //creating a new user object
                            User user = new User(
                                    userJson.getInt("id"),
                                    userJson.getString("username"),
                                    userJson.getString("email"),
                                    userJson.getString("gender"),
                                    userJson.getInt("user_counter")
                            );

                            //storing the user in shared preferences

SharedPrefManager.getInstance(getApplicationContext()).userLogin(user);

                            //starting the profile activity
                            finish();
                            startActivity(new Intent(getApplicationContext(),
WordpressPostActivity.class));
                        } else {
                            Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            },
            new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Toast.makeText(getApplicationContext(), error.getMessage(),
Toast.LENGTH_SHORT).show();
                }
            }) {
        @Override
        protected Map<String, String> getParams() throws AuthFailureError {
            Map<String, String> params = new HashMap<>();
            params.put("username", username);
            params.put("email", email);
            params.put("password", password);
            params.put("gender", gender);
            return params;
        }
    };


}
```

LoginActivity

```
package com.myclassstudio.album;

import androidx.appcompat.app.AppCompatActivity;


import android.content.Intent;


import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;


public class LoginActivity extends AppCompatActivity {


    String greeting = "Hello";

    EditText editTextUsername, editTextPassword;
    ProgressBar progressBar;
    ImageView image;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        if (SharedPrefManager.getInstance(this).isLoggedIn()) {
            finish();
            startActivity(new Intent(this, WordpressPostActivity.class));
        }

        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        editTextUsername = (EditText) findViewById(R.id.editTextUsername);
        editTextPassword = (EditText) findViewById(R.id.editTextPassword);
        image = (ImageView) findViewById(R.id.imageView);


        //if user presses on login
        //calling the method login
        findViewById(R.id.buttonLogin).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                userLogin();
```

```
            }
        });

        //if user presses on not registered
        findViewById(R.id.textViewRegister).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //open register screen
                finish();
                startActivity(new Intent(getApplicationContext(),
MainActivity.class));
            }
        });

    }

    private void userLogin() {
        //first getting the values
        final String username = editTextUsername.getText().toString();
        final String password = editTextPassword.getText().toString();

        //validating inputs
        if (TextUtils.isEmpty(username)) {
            editTextUsername.setError("Please enter your username");
            editTextUsername.requestFocus();
            return;
        }

        if (TextUtils.isEmpty(password)) {
            editTextPassword.setError("Please enter your password");
            editTextPassword.requestFocus();
            return;
        }

        //if everything is fine
        StringRequest stringRequest = new StringRequest(Request.Method.POST,
URLs.URL_LOGIN,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        progressBar.setVisibility(View.GONE);

                        try {
                            //converting response to json object
                            JSONObject obj = new JSONObject(response);

                            //if no error in response
                            if (!obj.getBoolean("error")) {
                                Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();

                                //getting the user from the response
                                JSONObject userJson = obj.getJSONObject("user");

                                //creating a new user object
                                User user = new User(
                                        userJson.getInt("id"),
                                        userJson.getString("username"),
                                        userJson.getString("email"),
                                        userJson.getString("gender"),
                                        userJson.getInt("user_counter")
                                );

                                //storing the user in shared preferences
```

```
SharedPrefManager.getInstance(getApplicationContext()).userLogin(user);

                                //starting the profile activity
                                finish();
                                startActivity(new
Intent(getApplicationContext(), ProfileActivity.class));
                            } else {
                                Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        Toast.makeText(getApplicationContext(),
error.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                })

            {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> params = new HashMap<>();
                params.put("username", username);
                params.put("password", password);
                return params;
            }
        };

        VolleySingleton.getInstance(this).addToRequestQueue(stringRequest);
    }

}
```

Java Comments

Comment will no execute by the program, the explain and note for programmer

Single-line comments start with two forward slashes (`//`).

Any text between `//` and the end of the line is ignored by Java (will not be executed).

This example uses a single-line comment before a line of code:

Java Variables

Variables are containers for storing data values.

In Java, there are different **types** of variables, for example:

- `String` - stores text, such as "Hello". String values are surrounded by double quotes
- `int` - stores integers (whole numbers), without decimals, such as 123 or -123
- `float` - stores floating point numbers, with decimals, such as 19.99 or -19.99

- `char` - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- `boolean` - stores values with two states: true or false

Declaring (Creating) Variables

```
private String username, email, gender;
```

Final Variables - Constant

However, you can add the `final` keyword if you don't want others (or yourself) to overwrite existing values (this will declare the variable as "final" or "constant", which means unchangeable and read-only):

Java Identifiers

All Java **variables** must be **identified** with **unique names and Case sensitive**. Cannot duplicate

JAVA != java

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or have meaning (age, sum, totalVolume).

Note: It is recommended to use meaning names to create understandable and maintainable code:

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs
- Names must begin with a letter
- Names should start with a lowercase letter and it cannot contain whitespace
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Reserved words (like Java keywords, such as `int` or `boolean`) cannot be used as names

```
int myNum = 5;               // Integer (whole number)
float myFloatNum = 5.99f;    // Floating point number
char myLetter = 'D';         // Character
boolean myBool = true;       // Boolean
String myText = "Hello";     // String
```

Data types are divided into two groups:

- Primitive data types - includes `byte`, `short`, `int`, `long`, `float`, `double`, `boolean` and `char`

- Non-primitive data types - such as String, Arrays and Classes (you will learn more about these in a later chapter)

Primitive Data Types

A primitive data type specifies the size and type of variable values, and it has no additional methods.

There are 8 primitive data types in Java:

byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Non-primitive

The main difference between **primitive** and **non-primitive** data types are:

- Primitive types are predefined (already defined) in Java. Non-primitive types are created by the programmer and is not defined by Java (except for `String`).

Examples of non-primitive types are Strings, Arrays, Classes, Interface, etc. You will learn more about these in a later chapter.

Java Type Casting

Type casting is when you assign a value of one primitive data type to another type.

```
progressBar = (ProgressBar) findViewById(R.id.progressBar);
editTextUsername = (EditText) findViewById(R.id.editTextUsername);
editTextPassword = (EditText) findViewById(R.id.editTextPassword);
image = (ImageView) findViewById(R.id.imageView);
```

Java Operators

Java divides the operators into the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bitwise operators

Operator	Name	Description	Example	Try it
+	Addition	Adds together two values	x + y	
-	Subtraction	Subtracts one value from another	x - y	
*	Multiplication	Multiplies two values	x * y	
/	Division	Divides one value from another	x / y	
%	Modulus	Returns the division remainder	x % y	
++	Increment	Increases the value of a variable by 1	++x	
--	Decrement	Decreases the value of a variable by 1	--x	

Operator	Name	Example	Try it
==	Equal to	x == y	
!=	Not equal	x != y	
>	Greater than	x > y	
<	Less than	x < y	
>=	Greater than or equal to	x >= y	
<=	Less than or equal to	x <= y	

Java Strings

Strings are used for storing text.

A `String` variable contains a collection of characters surrounded by double quotes:

```
String greeting = "Hello";
```

String Length

A String in Java is actually an object, which contain methods that can perform certain operations on strings. For example, the length of a string can be found with the `length()` method:

Special Characters

Because strings must be written within quotes, Java will misunderstand this string, and generate an error:

```
String txt = "We are the so-called "Vikings" from the north.";
```

The solution to avoid this problem, is to use the **backslash escape character**.

The backslash (\) escape character turns special characters into string characters:

Escape character	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

Java Booleans

Very often, in programming, you will need a data type that can only have one of two values, like:

- YES / NO
- ON / OFF
- TRUE / FALSE

For this, Java has a `boolean` data type, which can take the values `true` or `false`.

```
boolean isJavaFun = true;
boolean isFishTasty = false;
System.out.println(isJavaFun);     // Outputs true
System.out.println(isFishTasty);   // Outputs false
```

```
if (TextUtils.isEmpty(username)) {
    editTextUsername.setError("Please enter your username");
```

```
    editTextUsername.requestFocus();
    return;
}
```

Java Conditions and If Statements

Java supports the usual logical conditions from mathematics:

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

You can use these conditions to perform different actions for different decisions.

Java has the following conditional statements:

- Use `if` to specify a block of code to be executed, if a specified condition is true
- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false
- Use `switch` to specify many alternative blocks of code to be executed

The if Statement

Use the `if` statement to specify a block of Java code to be executed if a condition is `true`.

```
if (condition) {
        // block of code to be executed if the condition is true
    }

    int time = 20;
if (time < 18) {
        System.out.println("Good day.");
    } else {
        System.out.println("Good evening.");
    }
```

Short Hand If...Else (Ternary Operator)

```
variable = (condition) ? expressionTrue :  expressionFalse;
```

Java Switch Statements

Use the `switch` statement to select one of many code blocks to be executed.

```
switch(expression) {
        case x:
            // code block
            break;
        case y:
            // code block
            break;
        default:
            // code block
    }
```

```
switch (day) {
        case 1:
            System.out.println(“case 1");
            break;
        case 1:
            System.out.println(“case 2");
            break;
        default:
            System.out.println(“have good day");
    }
```

Java While Loop

The `while` loop loops through a block of code as long as a specified condition is `true`:

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

The Do/While Loop

The `do/while` loop is a variant of the `while` loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
int i = 0;
```

```
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

Java For Loop

When you know exactly how many times you want to loop through a block of code, use the `for` loop instead of a `while` loop:

```
for (int i = 0; i < 5; i++) {
        System.out.println(i);
    }
```

For-Each Loop

There is also a "**for-each**" loop, which is used exclusively to loop through elements in an **array**:

```
  String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
        System.out.println(i);
    }
```

Java Break

You have already seen the `break` statement used in an earlier chapter of this tutorial. It was used to "jump out" of a `switch` statement.

The `break` statement can also be used to jump out of a **loop**.

This example jumps out of the loop when i is equal to 4:

```
for (int i = 0; i < 10; i++) {
        if (i == 4) {
            break;
        }
        System.out.println(i);
    }
```

Java Continue

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of 4:

```
for (int i = 0; i < 10; i++) {
        if (i == 4) {
            continue;
        }
        System.out.println(i);
    }
```

Java Arrays

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with **square brackets**:

```
String[] cars;
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
```

Access the Elements of an Array

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
System.out.println(cars[0]);
```

Loop Through an Array

You can loop through the array elements with the `for` loop, and use the `length` property to specify how many times the loop should run.

The following example outputs all elements in the **cars** array:

Java Methods - like a Functions

A **method** is a function that contain some program to run.You pass data, known as parameters, into a method.

Methods are used to perform certain actions, and they are also known as **functions**.

Why use methods? To reuse code: define the code once, and use it many times.

Create a Method

```
public class MyClass {
  static void myMethod() {
    // code to be executed
  }
}
```

Call a Method

To call a method in Java, write the method's name followed by two parentheses **()** and a semicolon**;**

In the following example, `myMethod()` is used to print a text (the action), when it is called:

Parameters and Arguments

Information can be passed to methods as parameter. Parameters act as variables inside the method.

Parameters are specified after the method name, inside the parentheses. You can add as many parameters as you want, just separate them with a comma.

The following example has a method that takes a `String` called **fname** as parameter. When the method is called, we pass along a first name, which is used inside the method to print the full name:

```
public class MyClass {
    static void myMethod(String fname) {
        System.out.println(fname + " test");
    }

    public static void main(String[] args) {
        myClass.myMethod(“sdjnn");
        myClass.myMethod(“Jsngnny");
        myClass.myMethod(“ksjdfjs");
    }
}
```

Return Values

The `void` keyword, used in the examples above, indicates that the method should not return a value. If you want the method to return a value, you can use a primitive data type (such as `int`, `char`, etc.) instead of `void`, and use the `return` keyword inside the method:

```
public class MyClass {
    static int myMethod(int x) {
        return 5 + x;
    }

    public static void main(String[] args) {
        System.out.println(myMethod(3));
    }
}
```

Create a Class

To create a class, use the keyword `class`:

```
public class User {


    private int id, user_counter;
    private String username, email, gender;

    public User(int id, String username, String email, String gender, int user_counter)
{
        this.id = id;
        this.username = username;
        this.email = email;
        this.gender = gender;
        this.user_counter = user_counter;
    }

    public int getId() {
        return id;
    }

    public String getUsername() {
        return username;
    }

    public String getEmail() {
        return email;
    }

    public String getGender() {
        return gender;
    }

    public int getUserCounter(){
        return user_counter;
    }


    public void setUserCounter(int user_counter){

        this.user_counter= user_counter;
    }

    public void addUserCounter(){

        this.user_counter= this.user_counter + 1 ;
    }

}
```

Create an Object = new

```
MyClass myObj = new MyClass();
```

Java Class Attributes

In the previous chapter, we used the term "variable" for **x** in the example (as shown below). It is actually an **attribute** of the class. Or you could say that class attributes are variables within a class:

```
public class MyClass {
  int x = 5;
```

```
  int y = 3;
}
```

Static vs. Non-Static

You will often see Java programs that have either `static` or `public` attributes and methods.

In the example above, we created a `static` method, which means that it can be accessed without creating an object of the class, unlike `public`, which can only be accessed by objects:

```
public class MyClass {
    // Static method
    static void myStaticMethod() {
        System.out.println("Static methods can be called without creating
objects");
    }

    // Public method
    public void myPublicMethod() {
        System.out.println("Public methods must be called by creating objects");
    }

    // Main method
    public static void main(String[] args) {
        myStaticMethod(); // Call the static method
        // myPublicMethod(); This would compile an error

        MyClass myObj = new MyClass(); // Create an object of MyClass
        myObj.myPublicMethod(); // Call the public method on the object
    }
}
```

Access Methods With an Object

```java
// Create a Car class
public class Car {

    // Create a fullThrottle() method
    public void fullThrottle() {
        System.out.println("The car is going as fast as it can!");
    }

    // Create a speed() method and add a parameter
    public void speed(int maxSpeed) {
        System.out.println("Max speed is: " + maxSpeed);
    }

    // Inside main, call the methods on the myCar object
    public static void main(String[] args) {
        Car myCar = new Car();     // Create a myCar object
        myCar.fullThrottle();      // Call the fullThrottle() method
        myCar.speed(200);          // Call the speed() method
    }
}
```

Java Constructors

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes:

```
public User(int id, String username, String email, String gender,
int user_counter) {
    this.id = id;
    this.username = username;
    this.email = email;
    this.gender = gender;
    this.user_counter = user_counter;
}
```

Encapsulation and Access Modifiers

Modifier	Description	Try it
`public`	The code is accessible for all classes	
`private`	The code is only accessible within the declared class	
default	The code is only accessible in the same package. This is used when you don't specify a modifier. You will learn more about packages in the Packages chapter	
`protected`	The code is accessible in the same package and **subclasses**. You will learn more about subclasses and superclasses in the Inheritance chapter	

Non-Access Modifiers

Modifier	Description
`final`	Attributes and methods cannot be overridden/modified
`static`	Attributes and methods belongs to the class, rather than an object
`abstract`	Can only be used in an abstract class, and can only be used on methods. The method does not have a body, for example **abstract void run();**. The body is provided by the subclass (inherited from). You will learn more about inheritance and abstraction in the Inheritance and Abstraction chapters
`transient`	Attributes and methods are skipped when serializing the object containing them
`synchronized`	Methods can only be accessed by one thread at a time
`volatile`	The value of an attribute is not cached thread-locally, and is always read from the "main memory"

Modifier	Description	Try it
`final`	The class cannot be inherited by other classes (You will learn more about inheritance in the Inheritance chapter)	
`abstract`	The class cannot be used to create objects (To access an abstract class, it must be inherited from another class. You will learn more about inheritance and abstraction in the Inheritance and Abstraction chapters)	

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.length; i++) {
```

```
  System.out.println(cars[i]);
}
```

Final

If you don't want the ability to override existing attribute values, declare attributes as `final`:

Static

A `static` method means that it can be accessed without creating an object of the class, unlike `public`:

Abstract

An `abstract` method belongs to an `abstract` class, and it does not have a body. The body is provided by the subclass:

Encapsulation

The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users. To achieve this, you must:

- declare class variables/attributes as `private`
- provide public **get** and **set** methods to access and update the value of a `private` variable

Get and Set

You learned from the previous chapter that `private` variables can only be accessed within the same class (an outside class has no access to it). However, it is possible to access them if we provide public **get** and **set** methods.

The `get` method returns the variable value, and the `set` method sets the value.

Syntax for both is that they start with either `get` or `set`, followed by the name of the variable, with the first letter in upper case:

```
public class Person {
  private String name; // private = restricted access

  // Getter
  public String getName() {
```

```
    return name;
  }

  // Setter
  public void setName(String newName) {
    this.name = newName;
  }
}
```

Why Encapsulation?

- Better control of class attributes and methods
- Class attributes can be made **read-only**
- Flexible: the programmer can change one part of the code without affecting other parts
- Increased security of data

Java Packages & API

- Built-in Packages (packages from the Java API)
- User-defined Packages (create your own packages)

Import a Class

If you find a class you want to use, for example, the `Scanner` class, **which is used to get user input**, write the following code:

```
import java.util.Scanner;
```

In the example above, `java.util` is a package, while `Scanner` is a class of the `java.util` package.

```
import android.content.Intent;
import android.view.View;
import android.text.TextUtils;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ProgressBar;
import android.widget.RadioButton;
import android.widget.RadioGroup;
```

Import a Package

There are many packages to choose from. In the previous example, we used the `Scanner` class from the `java.util` package. This package also contains date and time facilities, random-number generator and other utility classes.

To import a whole package, end the sentence with an asterisk sign (`*`). The following example will import ALL the classes in the `java.util` package:

Example

```
import java.util.*;
```

Java Inheritance

Java Inheritance (child and parent) - extends

In Java, it is possible to inherit attributes and methods from one class to another. We group the "inheritance concept" into two categories:

- **child** - the class that inherits from another class
- **parent** - the class being inherited from

To inherit from a class, use the `extends` keyword.

In the example below, the `Car` class (subclass) inherits the attributes and methods from the `Vehicle` class (superclass):

```
public class MainActivity extends AppCompatActivity {
```

Java Polymorphism

Java Polymorphism

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

```
class Human {
    public void humanSing() {
        System.out.println("The human voice");
    }
}

class male extends Human {
    public void humanSing() {
        System.out.println(“Low voice");
    }
}

class female extends Human {
    public void humanSing() {
        System.out.println(“High Voice");
    }
}
```

Java Date and Time

Class	Description
`LocalDate`	Represents a date (year, month, day (yyyy-MM-dd))
`LocalTime`	Represents a time (hour, minute, second and milliseconds (HH-mm-ss-zzz))
`LocalDateTime`	Represents both a date and a time (yyyy-MM-dd-HH-mm-ss.zzz)
`DateTimeFormatter`	Formatter for displaying and parsing date-time objects

Java ArrayList

- Vector no limit of Length of Array
- Hashtable - use (get) and (put). (key, val) put in container
- Stackwith push(), pop() methods

Java ArrayList

The `ArrayList` class is a resizable array, which can be found in the `java.util` package.

```
import java.util.ArrayList; // import the ArrayList class

    ArrayList<String> cars = new ArrayList<String>(); // Create an ArrayList object
```

```
import java.util.ArrayList;

    public class MyClass {
        public static void main(String[] args) {
            ArrayList<String> cars = new ArrayList<String>();
            cars.add(“car 1");
            cars.add("car 2");
            cars.add("car 3");
            cars.add("car 4");
            System.out.println(cars);
        }
    }
```

Java HashMap

Java HashMap

You learned from the previous chapter, that Arrays store items as an ordered collection, and you have to access them with an index number (`int` type). A `HashMap` however, store items in "**key**/**value**" pairs, and you can access them by an index of another type (e.g. a `String`).

One object is used as a key (index) to another object (value). It can store different types: `String` keys and `Integer` values, or the same type, like: `String` keys and `String` values:

```
import java.util.HashMap; // import the HashMap class

HashMap<String, String> capitalCities = new HashMap<String,
String>();
```

Add Items

```
// Import the HashMap class
import java.util.HashMap;

    public class MyClass {
        public static void main(String[] args) {
            // Create a HashMap object called capitalCities
            HashMap<String, String> capitalCities = new HashMap<String,
String>();

            // Add keys and values (Country, City)

            capitalCities.put("USA", "Washington DC");
            System.out.println(capitalCities);
        }
    }
```

Java Exceptions - Try...Catch

Java Exceptions

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things.

When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an **exception** (throw an error).

Java try and catch MultiThread

The `try` statement allows you to define a block of code to be tested for errors while it is being executed.

The `catch` statement allows you to define a block of code to be executed, if an error occurs in the try block.

The `try` and `catch` keywords come in pairs:

MultiThread

Java is a *multi-threaded programming language* which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

By definition, multitasking is when multiple processes share common processing resources such as a CPU. Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program.

```
uploadButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {


            dialog = ProgressDialog.show(CoverActivity.this, "", "Uploading
file...", true);

            new Thread(new Runnable() {
                public void run() {
                    runOnUiThread(new Runnable() {
```

```
                    public void run() {
                        messageText.setText("uploading started.....");
                    }
                });

                path = getPath(filePath);

                    uploadFile(path);

            }
        }).start();

        startActivity(new Intent(getApplicationContext(),
ProfileActivity.class));


    }
});
```

Life Cycle of a Thread

A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. The following diagram shows the complete life cycle of a thread.

Following are the stages of the life cycle –

- **New** – A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a **born thread**.

- **Runnable** – After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

- **Waiting** – Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.

- **Timed Waiting** – A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

- **Terminated (Dead)** – A runnable thread enters the terminated state when it completes its task or otherwise terminates.

```
class RunnableDemo implements Runnable {
    private Thread t;
    private String threadName;

    RunnableDemo( String name) {
        threadName = name;
        System.out.println("Creating " +  threadName );
    }

    public void run() {
```

```
        System.out.println("Running " +  threadName );
        try {
            for(int i = 4; i > 0; i--) {
                System.out.println("Thread: " + threadName + ", " + i);
                // Let the thread sleep for a while.
                Thread.sleep(50);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread " +  threadName + " interrupted.");
        }
        System.out.println("Thread " +  threadName + " exiting.");
    }

    public void start () {
        System.out.println("Starting " +  threadName );
        if (t == null) {
            t = new Thread (this, threadName);
            t.start ();
        }
    }
}

public class TestThread {

    public static void main(String args[]) {
        RunnableDemo R1 = new RunnableDemo( "Thread-1");
        R1.start();

        RunnableDemo R2 = new RunnableDemo( "Thread-2");
        R2.start();
    }
}
```

```
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Thread: Thread-1, 4
Running Thread-2
Thread: Thread-2, 4
Thread: Thread-1, 3
Thread: Thread-2, 3
Thread: Thread-1, 2
Thread: Thread-2, 2
Thread: Thread-1, 1
Thread: Thread-2, 1
Thread Thread-1 exiting.
Thread Thread-2 exiting.
```

Java Reserved Keywords

Java has a set of keywords that are reserved words that cannot be used as variables, methods, classes, or any other identifiers:

abstract	do	if	private	this
assert	double	implements	protected	throw
boolean	else	import	public	throws
break	enum	instanceof	return	transient
byte	extends	int	short	true
case	false	interface	static	try
catch	final	long	strictfp	void
char	finally	native	super	volatile
class	float	new	switch	while
const	for	null	synchronized	continue
default	goto	package		

Keyword - Clear Concept

Package
Extends inheritance
Hashmap
Bundle
Super this call inheritance instance
Findviewbyid R
Casting
Setonckicklistener
IsEmpty
StringRequest
Jasonobject
Startactivity intent
Appcompactactivity
User class object
Upload photo multiparty
Multi thread try catch
Choose file
Findview by I'd R text field
Get set
Constructor
Res file layout value
Manifest xml
Grade
Showfilechoose
Getpermission
Change refactor package name
Requestqueue
Layout xml
Httprespone
Instanceof
Override
Abstract -hiding info
Static - only one value even create many object
Polymorphism
https://antrash.pixnet.net/blog/post/69452213-%E8%AB%96%E7%89%A9%E4%BB%B6%E5%B0%8E%E5%90%91part-5%EF%BC%9Apolymorphism
Interface inheritance
https://www.itread01.com/p/1449114.html

Simlulator
Tool device file explorer

PHP Tutorial

What Do I Need?

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Basic PHP Syntax

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

```
<?php
// PHP code goes here
?>
```

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Please check out our PHP reference for a complete overview of the PHP built-in functions.

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

```
<?php
function writeMsg() {
   echo "Hello world!";
}

writeMsg(); // call the function
?>
```

The PHP superglobals $_GET and $_POST are used to collect form-data.

PHP - A Simple HTML Form - POST GET

Post - send more important message like password, can be longer value

Get - can view in the link/url http://domain.com?username=xxxx&gender=male

The example below displays a simple HTML form with two input fields and a submit button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

PHP Form Validation

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

Text Fields

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

Gender:

```
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
```

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

PHP - Keep The Values in The Form

To show the values in the input fields after the user hits the submit button, we add a little PHP script inside the value attribute of the following input fields: name, email, and website. In the comment textarea field, we put the script between the <textarea> and </textarea> tags. The little script outputs the value of the $name, $email, $website, and $comment variables.

Then, we also need to show which radio button that was checked. For this, we must manipulate the checked attribute (not the value attribute for radio buttons):

Name: <input type="text" name="name" value="<?php echo $name;?>">

E-mail: <input type="text" name="email" value="<?php echo $email;?>">

Website: <input type="text" name="website" value="<?php echo $website;?>">

Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>

Gender:
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="male") echo "checked";?>
value="male">Male
<input type="radio" name="gender"
<?php if (isset($gender) && $gender=="other") echo "checked";?>
value="other">Other

PHP Include Files

PHP include and require Statements

It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

- `require` will produce a fatal error (E_COMPILE_ERROR) and stop the script
- `include` will only produce a warning (E_WARNING) and the script will continue

PHP File Upload

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the `file_uploads` directive, and set it to On:

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
// Check file size
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
```

```
    echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
    $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not uploaded.";
// if everything is ok, try to upload file
} else {
    if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
        echo "The file ". basename( $_FILES["fileToUpload"]["name"]). " has been uploaded.";
    } else {
        echo "Sorry, there was an error uploading your file.";
    }
}
?>
```

PHP Sessions

What is a PHP Session? e.g. Store Username

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

```
<?php
// Start the session
session_start();
?>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>
```

What is JSON?

JSON stands for JavaScript Object Notation, and is a syntax for storing and exchanging data.

Since the JSON format is a text-based format, it can easily be sent to and from a server, and used as a data format by any programming language.

PHP and JSON

PHP has some built-in functions to handle JSON.

First, we will look at the following two functions:

- json_encode()
- json_decode()

```
<?php
$age = array("Peter"=>35, "Ben"=>37, "Joe"=>43);

echo json_encode($age);
?>
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

var_dump(json_decode($jsonobj));
?>
```

User API userapi.php

```
<?php

    require_once 'dbConnect.php';

    //an array to display response
    $response = array();

    //if it is an api call
    //that means a get parameter named api call is set in the URL
    //and with this parameter we are concluding that it is an api call

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

        switch($_GET['apicall']){

            case 'signup':
                //checking the parameters required are available or not
                //
if(isTheseParametersAvailable(array('username','email','password'))){
```

```
                if(isset($_POST['username']) && !empty($_POST['username']) &&
isset($_POST['email']) && !empty($_POST['email']) && isset($_POST['password'])
&& !empty($_POST['password'])   ){

                    //getting the values
                    $username = strtolower($_POST['username']);
                    $email = strtolower($_POST['email']);
                    $password_str =   $_POST['password'];
                    $password = md5($_POST['password']);
                    $gender = $_POST['gender'];



                    //checking if the user is already exist with this
username or email
                    //as the email and username should be unique for every
user
                    $stmt = $conn->prepare("SELECT id FROM submitimage_users
WHERE username = ? OR email = ?");
                    $stmt->bind_param("ss", $username, $email);
                    $stmt->execute();
                    $stmt->store_result();

                    //if the user already exist in the database
                    if($stmt->num_rows > 0){
                        $response['error'] = true;
                        $response['message'] = 'User already registered';
                        $stmt->close();
                    }else{

                        //if user is new creating an insert query
                        $stmt = $conn->prepare("INSERT INTO
submitimage_users (username, email, password, password_str, gender) VALUES
(?, ?, ?, ?, ?)");
                        $stmt->bind_param("sssss", $username, $email,
$password, $password_str, $gender);

                        //if the user is successfully added to the database
                        if($stmt->execute()){

                            //fetching the user back
                            $stmt = $conn->prepare("SELECT id, id, username,
email, gender FROM submitimage_users WHERE username = ?");
                            $stmt->bind_param("s",$username);
                            $stmt->execute();
                            $stmt->bind_result($userid, $id, $username,
$email, $gender);
                            $stmt->fetch();

                            $user = array(
                                'id'=>$id,
                                'username'=>$username,
                                'email'=>$email,
                                'gender'=>$gender
                            );

                            $stmt->close();

                            //adding the user data in response
                            $response['error'] = false;
                            $response['message'] = 'User registered
successfully';
                            $response['user'] = $user;
                        }
                    }
```

```
                }else{
                    $response['error'] = true;
                    $response['message'] = 'Required parameters are not
available';
                }

            break;

            case 'login':
                //for login we need the username and password
                //if(isTheseParametersAvailable(array('username',
'password'))){

                if(isset($_POST['username']) && !empty($_POST['username']) &&
isset($_POST['password']) && !empty($_POST['password'])  ){

                    //getting values
                    $username = $_POST['username'];
                    $password = md5($_POST['password']);

                    //creating the query
                    $stmt = $conn->prepare("SELECT id, username, email,
gender, user_counter FROM submitimage_users WHERE username = ? AND password
= ?");
                    $stmt->bind_param("ss",$username, $password);

                    $stmt->execute();

                    $stmt->store_result();

                    //if the user exist with given credentials
                    if($stmt->num_rows > 0){

                        $stmt->bind_result($id, $username, $email, $gender,
$user_counter);
                        $stmt->fetch();

                        $user = array(
                            'id'=>$id,
                            'username'=>$username,
                            'email'=>$email,
                            'gender'=>$gender,
                                'user_counter'=>$user_counter
                        );

                        $response['error'] = false;
                        $response['message'] = 'Login successfull';
                        $response['user'] = $user;
                    }else{
                        //if the user not found
                        $response['error'] = true;
                        $response['message'] = 'Invalid username or
password';
                    }
                }
            break;

            default:
                $response['error'] = true;
                $response['message'] = 'Invalid Operation Called';
        }

    }else{
        //if it is not api call
        //pushing appropriate values to response array
        $response['error'] = true;
```

```
        $response['message'] = 'Invalid API Call';
    }

    //displaying the response in json structure
    echo json_encode($response);

    //function validating all the paramters are available
    //we will pass the required parameters to this function
    function isTheseParametersAvailable($params){

        //traversing through all the parameters
        foreach($params as $param){
            //if the paramter is not available
            if(!isset($_POST[$param])){
                //return false
                return false;
            }
        }
        //return true if every param is available
        return true;
    }

?>
```

Open a Connection to MySQL

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

```php
echo "Connected successfully";
?>
```

A database consists of one or more tables.

You will need special CREATE privileges to create or to delete a MySQL database.

Create a MySQL Database Using MySQLi and PDO

The CREATE DATABASE statement is used to create a database in MySQL.

The following examples create a database named "myDB":

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

Create a MySQL Table Using MySQLi and PDO

The CREATE TABLE statement is used to create a table in MySQL.

We will create a table named "MyGuests", with five columns: "id", "firstname", "lastname", "email" and "reg_date":

PHP MySQL Insert Data

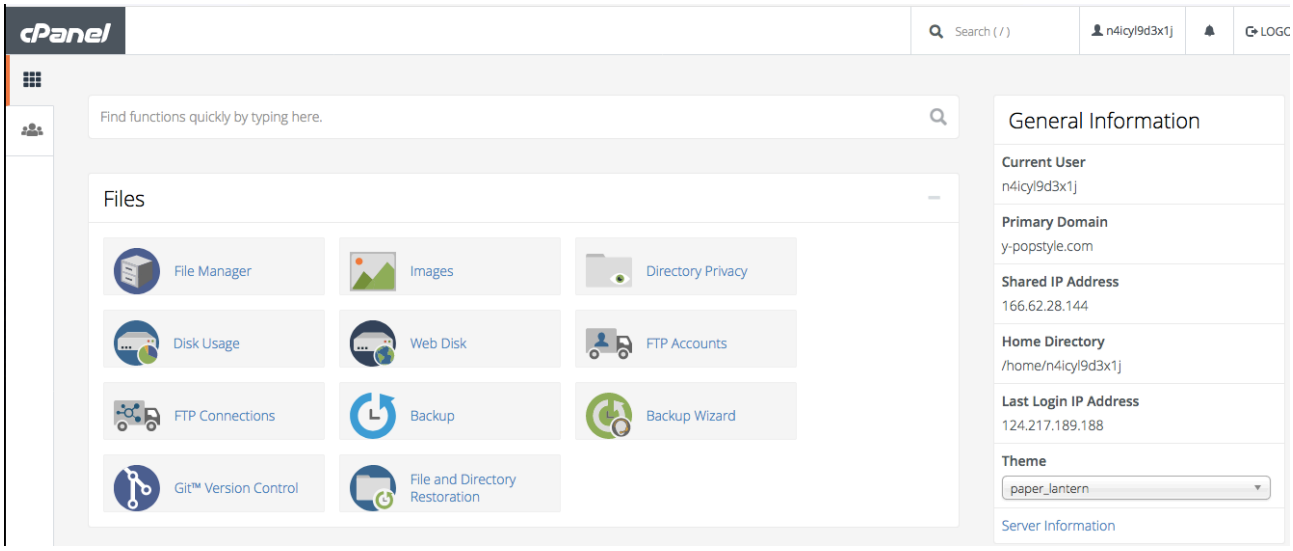
```
  CREATE TABLE `submitimage_users` (
            `id` int(11) NOT NULL,
  `username` varchar(200) NOT NULL,
  `email` varchar(200) NOT NULL,
  `password` text NOT NULL,
            `password_str` varchar(100) DEFAULT NULL,
  `first_name` varchar(50) DEFAULT NULL,
  `last_name` varchar(50) DEFAULT NULL,
  `description` varchar(100) DEFAULT NULL,
  `dateofbirth` date NOT NULL,
            `gender` varchar(6) NOT NULL,
  `user_counter` int(20) NOT NULL DEFAULT '0',
            `admin` int(10) NOT NULL DEFAULT '0',
            `trainer` int(10) NOT NULL DEFAULT '0',
            `registerdate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

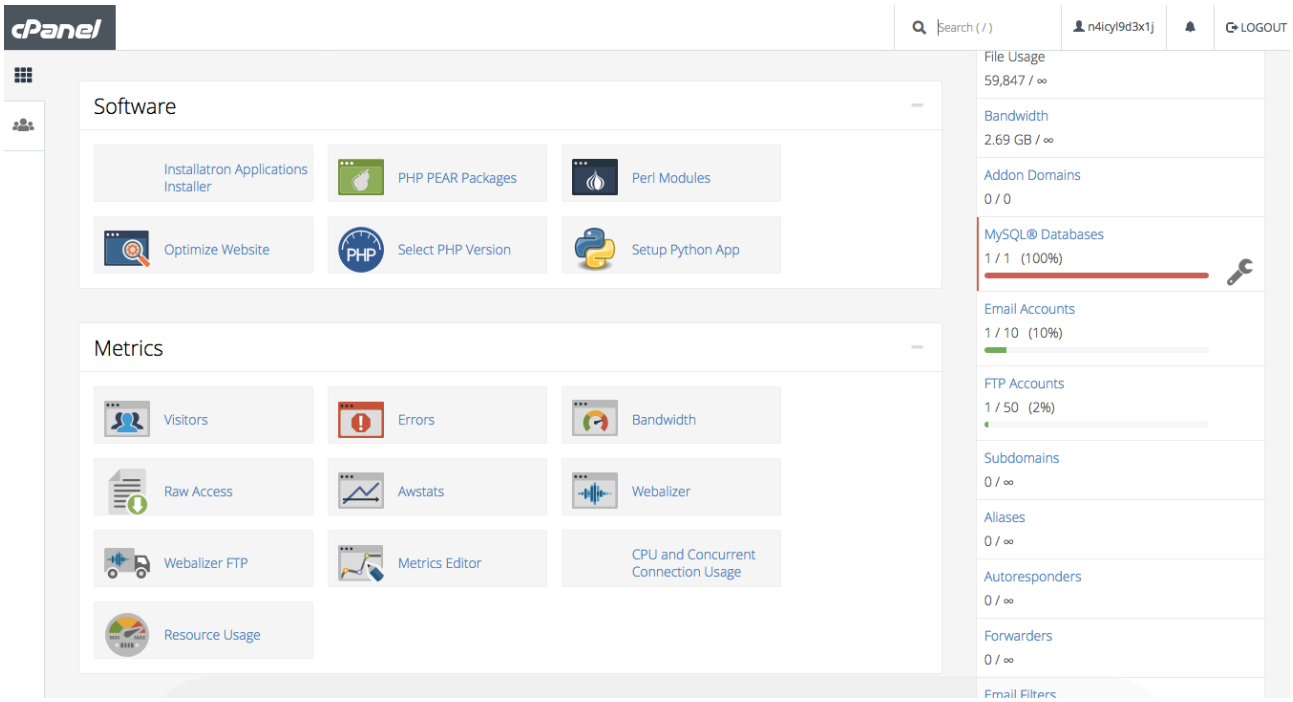
MySQL Functions

Select Distinct where and or note order by insert into update delete select top, count, LIKE, wildcards, joins, inner join , left join , right join , full join group by

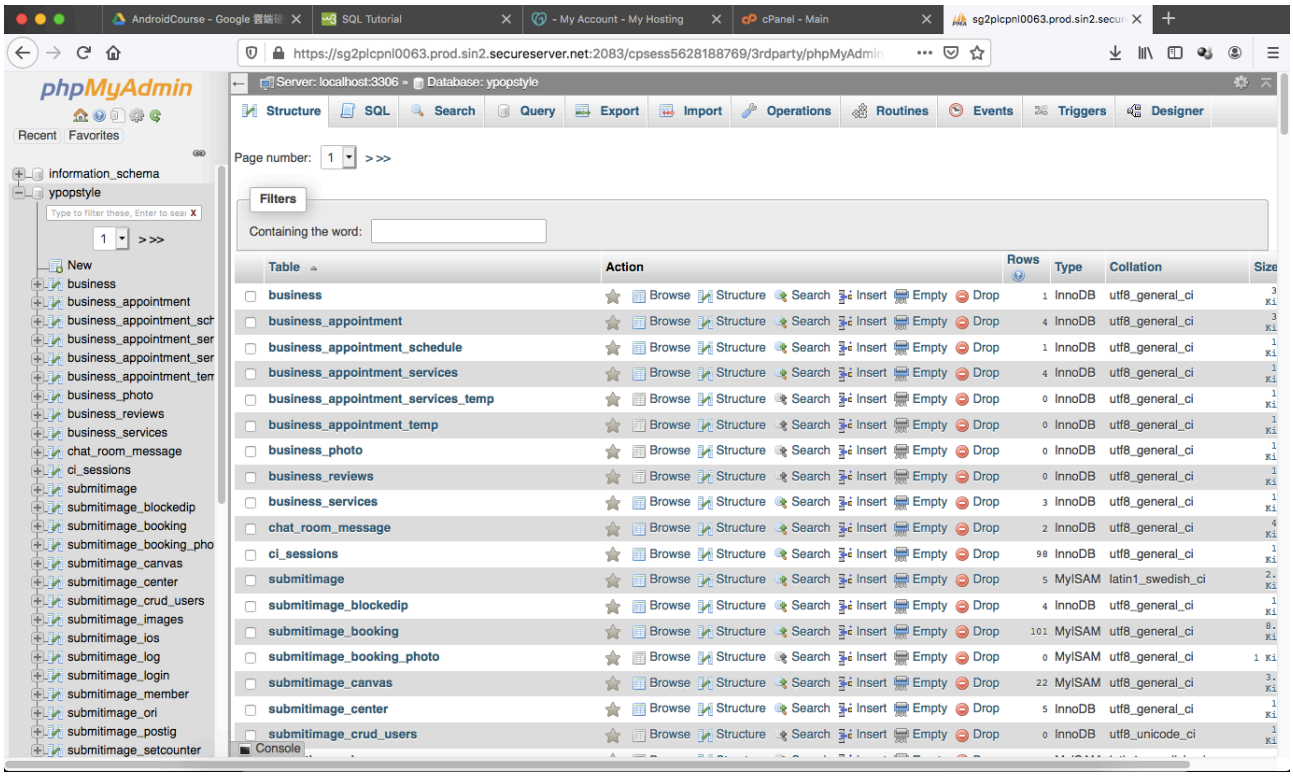
Cpanel

FTP File Transfer, Folder, Email, Phpmyadmin, Domain

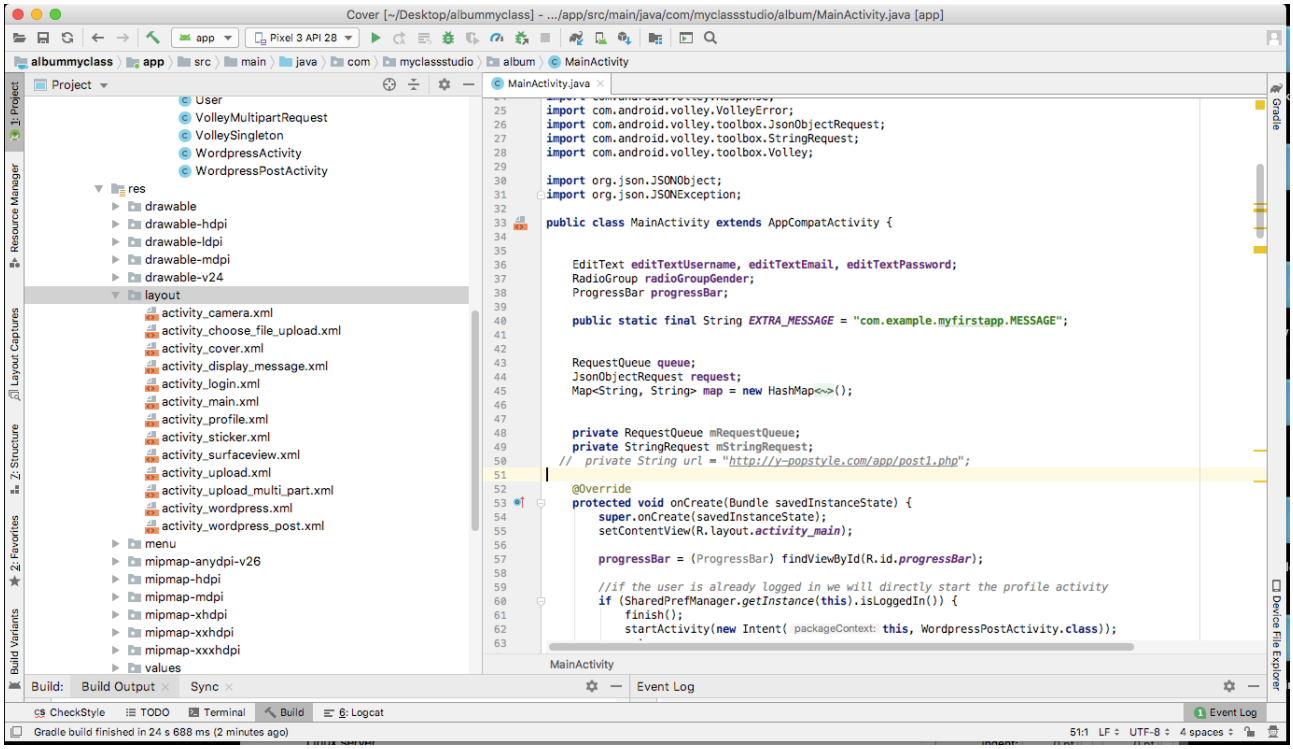




phpMyAdmin



Android Studio



```java
package com.myclassstudio.album;

import androidx.appcompat.app.AppCompatActivity;


import android.content.Intent;


import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;


public class LoginActivity extends AppCompatActivity {


    String greeting = "Hello";

    EditText editTextUsername, editTextPassword;
    ProgressBar progressBar;
    ImageView image;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        if (SharedPrefManager.getInstance(this).isLoggedIn()) {
            finish();
            startActivity(new Intent(this, WordpressPostActivity.class));
        }

        progressBar = (ProgressBar) findViewById(R.id.progressBar);
        editTextUsername = (EditText) findViewById(R.id.editTextUsername);
        editTextPassword = (EditText) findViewById(R.id.editTextPassword);
        image = (ImageView) findViewById(R.id.imageView);


        //if user presses on login
        //calling the method login
        findViewById(R.id.buttonLogin).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                userLogin();
            }
        });

        //if user presses on not registered
        findViewById(R.id.textViewRegister).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //open register screen
                finish();
                startActivity(new Intent(getApplicationContext(), MainActivity.class));
```

```
            }
        });

    }

    private void userLogin() {
        //first getting the values
        final String username = editTextUsername.getText().toString();
        final String password = editTextPassword.getText().toString();

        //validating inputs
        if (TextUtils.isEmpty(username)) {
            editTextUsername.setError("Please enter your username");
            editTextUsername.requestFocus();
            return;
        }

        if (TextUtils.isEmpty(password)) {
            editTextPassword.setError("Please enter your password");
            editTextPassword.requestFocus();
            return;
        }

        //if everything is fine
        StringRequest stringRequest = new StringRequest(Request.Method.POST,
URLs.URL_LOGIN,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        progressBar.setVisibility(View.GONE);

                        try {
                            //converting response to json object
                            JSONObject obj = new JSONObject(response);

                            //if no error in response
                            if (!obj.getBoolean("error")) {
                                Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();

                                //getting the user from the response
                                JSONObject userJson = obj.getJSONObject("user");

                                //creating a new user object
                                User user = new User(
                                        userJson.getInt("id"),
                                        userJson.getString("username"),
                                        userJson.getString("email"),
                                        userJson.getString("gender"),
                                        userJson.getInt("user_counter")
                                );

                                //storing the user in shared preferences

SharedPrefManager.getInstance(getApplicationContext()).userLogin(user);

                                //starting the profile activity
                                finish();
                                startActivity(new Intent(getApplicationContext(),
ProfileActivity.class));
                            } else {
                                Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
```

```
                        Toast.makeText(getApplicationContext(), error.getMessage(),
Toast.LENGTH_SHORT).show();
                    }
                })

            {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> params = new HashMap<>();
                params.put("username", username);
                params.put("password", password);
                return params;
            }
        };

        VolleySingleton.getInstance(this).addToRequestQueue(stringRequest);
    }

}
```

```
package com.myclassstudio.album;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.content.Intent;
import android.view.View;
import android.text.TextUtils;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.ProgressBar;
import android.widget.RadioButton;
import android.widget.RadioGroup;


import java.util.HashMap;
import java.util.Map;


import com.android.volley.AuthFailureError;
import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;

import org.json.JSONObject;
import org.json.JSONException;

public class MainActivity extends AppCompatActivity {


    EditText editTextUsername, editTextEmail, editTextPassword;
    RadioGroup radioGroupGender;
    ProgressBar progressBar;

    public static final String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";


    RequestQueue queue;
    JsonObjectRequest request;
    Map<String, String> map = new HashMap<String, String>();
```

```
    private RequestQueue mRequestQueue;
    private StringRequest mStringRequest;
  //  private String url = "http://y-popstyle.com/app/post1.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        progressBar = (ProgressBar) findViewById(R.id.progressBar);

        //if the user is already logged in we will directly start the profile
activity
        if (SharedPrefManager.getInstance(this).isLoggedIn()) {
            finish();
            startActivity(new Intent(this, WordpressPostActivity.class));
            return;
        }

        editTextUsername = (EditText) findViewById(R.id.editTextUsername);
        editTextEmail = (EditText) findViewById(R.id.editTextEmail);
        editTextPassword = (EditText) findViewById(R.id.editTextPassword);
        radioGroupGender = (RadioGroup) findViewById(R.id.radioGender);


        findViewById(R.id.buttonRegister).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //if user pressed on button register
                //here we will register the user to server
                registerUser();
            }
        });

        findViewById(R.id.textViewLogin).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //if user pressed on login
                //we will open the login screen
                finish();
                startActivity(new Intent(MainActivity.this,
LoginActivity.class));
            }
        });
    }

    private void registerUser() {
        final String username = editTextUsername.getText().toString().trim();
        final String email = editTextEmail.getText().toString().trim();
        final String password = editTextPassword.getText().toString().trim();

        final String gender = ((RadioButton)
findViewById(radioGroupGender.getCheckedRadioButtonId())).getText().toString();

        //first we will do the validations

        if (TextUtils.isEmpty(username)) {
            editTextUsername.setError("Please enter username");
            editTextUsername.requestFocus();
            return;
        }
```

```
        if (TextUtils.isEmpty(email)) {
            editTextEmail.setError("Please enter your email");
            editTextEmail.requestFocus();
            return;
        }

        if (!android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
            editTextEmail.setError("Enter a valid email");
            editTextEmail.requestFocus();
            return;
        }

        if (TextUtils.isEmpty(password)) {
            editTextPassword.setError("Enter a password");
            editTextPassword.requestFocus();
            return;
        }

        StringRequest stringRequest = new StringRequest(Request.Method.POST,
URLs.URL_REGISTER,
                new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        progressBar.setVisibility(View.GONE);

                        try {
                            //converting response to json object
                            JSONObject obj = new JSONObject(response);

                            //if no error in response
                            if (!obj.getBoolean("error")) {
                                Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();

                                //getting the user from the response
                                JSONObject userJson = obj.getJSONObject("user");

                                //creating a new user object
                                User user = new User(
                                        userJson.getInt("id"),
                                        userJson.getString("username"),
                                        userJson.getString("email"),
                                        userJson.getString("gender"),
                                        userJson.getInt("user_counter")
                                );

                                //storing the user in shared preferences

SharedPrefManager.getInstance(getApplicationContext()).userLogin(user);

                                //starting the profile activity
                                finish();
                                startActivity(new
Intent(getApplicationContext(), WordpressPostActivity.class));
                            } else {
                                Toast.makeText(getApplicationContext(),
obj.getString("message"), Toast.LENGTH_SHORT).show();
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                },
                new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
```

```
                        Toast.makeText(getApplicationContext(),
error.getMessage(), Toast.LENGTH_SHORT).show();
                    }
                }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> params = new HashMap<>();
                params.put("username", username);
                params.put("email", email);
                params.put("password", password);
                params.put("gender", gender);
                return params;
            }
        };

        VolleySingleton.getInstance(this).addToRequestQueue(stringRequest);

    }

    /** Called when the user taps the Send button */
    public void sendMessage(View view) {
        // Do something in response to button
        //Intent intent = new Intent(this, DisplayMessageActivity.class);
        //EditText editText = (EditText) findViewById(R.id.editText);
        //String message = editText.getText().toString();
        //intent.putExtra(EXTRA_MESSAGE, message);
        //startActivity(intent);

        // the request queue
        queue = Volley.newRequestQueue(this);

        // the parameters for the php
        // map.put(KEY, VALUE);
        map.put("name", "test");

        // the JSON request
        // JsonObjectRequest(METHOD, URL, JSONOBJECT(PARAMETERS), OK_LISTENER,
ERROR_LISTENER);
        request = new JsonObjectRequest(
                Request.Method.POST, // the request method
                "http://y-popstyle.com/album/postjoson.php", // the URL
                new JSONObject(map), // the parameters for the php
                new Response.Listener<JSONObject>() { // the response listener
                    @Override
                    public void onResponse(JSONObject response){
                        // here you parse the json response
                    }
                },
                new Response.ErrorListener() { // the error listener
                    @Override
                    public void onErrorResponse(VolleyError error) {

                      /* here you can warn the user that there
                      was an error while trying to get the json
                      information from the php  */
                    }
                });

        // executing the quere to get the json information
        queue.add(request);



        /*
        EditText editText = (EditText) findViewById(R.id.editText);
```

```
        String message = editText.getText().toString();
        url = "http://y-popstyle.com/app/post1.php";
        StringRequest stringRequest = new StringRequest(Request.Method.POST,
url,   new Response.Listener<String>() {
            @Override
            public void onResponse(String s) {

            }
        }, new Response.ErrorListener() {

            @Override
            public void onErrorResponse(VolleyError volleyError) {

            }
        }) {
            @Override
            protected Map<String, String> getParams() throws AuthFailureError {
                Map<String, String> map = new HashMap<String, String>();
                map.put("name", "test1");
                map.put("email", "test1");

                return map;
            }
        };
        */




    }



}
```

```java
package com.myclassstudio.album;

public class User {




    private int id, user_counter;
    private String username, email, gender;

    public User(int id, String username, String email, String gender, int
user_counter) {
        this.id = id;
        this.username = username;
        this.email = email;
        this.gender = gender;
        this.user_counter = user_counter;
    }

    public int getId() {
        return id;
    }

    public String getUsername() {
        return username;
    }

    public String getEmail() {
        return email;
    }

    public String getGender() {
        return gender;
    }

    public int getUserCounter(){
        return user_counter;
    }


    public void setUserCounter(int user_counter){

        this.user_counter= user_counter;
    }

    public void addUserCounter(){

        this.user_counter= this.user_counter + 1 ;
    }

}


package com.myclassstudio.album;

import androidx.appcompat.app.AppCompatActivity;


import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class ProfileActivity extends AppCompatActivity {
```

```
    TextView textViewId, textViewUsername, textViewEmail, textViewGender;


    // choose Sticker or Overlay Camera
    // counter for limit upload per month

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);

        //if the user is not logged in
        //starting the login activity
        if (!SharedPrefManager.getInstance(this).isLoggedIn()) {
            finish();
            startActivity(new Intent(this, LoginActivity.class));
        }


        textViewId = (TextView) findViewById(R.id.textViewId);
        textViewUsername = (TextView) findViewById(R.id.textViewUsername);
        textViewEmail = (TextView) findViewById(R.id.textViewEmail);
        textViewGender = (TextView) findViewById(R.id.textViewGender);


        //getting the current user
        User user = SharedPrefManager.getInstance(this).getUser();

        //setting the values to the textviews
        textViewId.setText(String.valueOf(user.getId()));
        textViewUsername.setText(user.getUsername());
        textViewEmail.setText(user.getEmail());
        textViewGender.setText(user.getGender());

        //when the user presses logout button
        //calling the logout method
        findViewById(R.id.buttonLogout).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
                SharedPrefManager.getInstance(getApplicationContext()).logout();
            }
        });

        findViewById(R.id.buttonUpload).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
              //  finish();
                startActivity(new Intent(getApplicationContext(),
CoverActivity.class));

            }
        });




    }
}
```