프레퍼런스 : 사용자가 입력한 정보, 환경 설정 정보 등을 간단히 저장 관리 할 수 있음

제일 단순한 저장 형태

각 애플리케이션에 고유한 설정값을 지정

<키, 값>의 조합으로 데이터 저장

θ 값에 이름을 부여하여 저장 θ 환경설정에 유용

Spref = getSharedPreferences("gpref", MODE\_PRIVATE);

// "gpref"라는 이름의 SharedPreferences 객체를 가져옵니다.

// MODE\_PRIVATE는 이 파일이 이 애플리케이션에서만 사용됨을 의미합니다.

editor = spref.edit();

// SharedPreferences 객체를 수정하기 위한 Editor 객체를 가져옵니다.

String temp1 = spref.getString("editText01", "1");

// "editText01" 키로 저장된 문자열 값을 가져와 temp1 변수에 저장합니다.

// 값이 없으면 기본값으로 "1"을 반환합니다.

edit01.setText(temp1);

// edit01 EditText 뷰에 temp1 값을 설정합니다.

editor.putString("editText01", txt01);

// txt01 문자열 값을 "editText01" 키로 SharedPreferences에 저장하도록 설정합니다.

editor.apply();

// 설정된 변경 사항을 비동기적으로 저장합니다.

파일 저장 : 디바이스나 저장 매체에 직접적으로 파일을 저장

μ데이터 읽기 θ Context.openFileInput() 메소드 호출, FileInputStream오 브젝트를 얻어서 처리

μ데이터 쓰기 θ Context.openFileOutput() 메소드 호출, FileOutputStream 오브젝트를 얻어서 처리

android:orientation = “horizontal” ◊ 가로 기준선을 생성

android:orientation = “vertical” ◊ 세로 기준선을 생성

비율로 지정하는 경우 : layout\_constraintGuide\_percent

시작점을 기준으로 거리를 지정하는 경우 : layout\_constraintGuide\_begin

끝점을 기준으로 거리를 지정하는 경우 : layout\_constraintGuide\_end

기본 이벤트 종류에 따른 처리 과정 학습

클릭

OnClickListener

void onClick(View v)

포커스 변경

OnFocusChangeListener

void onFocusChange(View v, boolean hasFocus)

롱클릭

OnLongClickListener

boolean onLongClick(View v)

키 (특정 위치를 사용하는 경우)

OnKeyListener

boolean onKey(View v, int keyCode, KeyEvent event)

키 (특정 위치를 사용하지 않는 경우)

onKeyDown(int keycode, KeyEvent event)

터치

OnTouchListener

boolean onTouch(View v, MotionEvent event)

KEYCODE\_0 ~ KEYCODE\_9 숫자 0 ~ 9

KEYCODE\_A ~ KEYCODE\_Z 알파벳 A ~ Z

WebView

θ WebKit 엔진 이용 θ 웹페이지를 화면에 디스플레이하는 뷰

AndroidManifest.xml

<uses-permission android:name="android.permission.INTERNET"/>

activity\_main.xml 레이아웃 파일에 WebView를 추가

<WebView android:id="@+id/webview" android:layout\_width="match\_parent" android:layout\_height="match\_parent" />

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// 레이아웃에서 WebView를 찾습니다.

WebView webView = findViewById(R.id.webview);

// WebView 설정을 가져옵니다.

WebSettings webSettings = webView.getSettings();

// JavaScript를 사용하도록 설정합니다.

webSettings.setJavaScriptEnabled(true);

// WebView 클라이언트를 설정하여 웹 페이지가 WebView에서 열리도록 합니다.

webView.setWebViewClient(new WebViewClient());

// 웹 페이지를 로드합니다.

webView.loadUrl("https://www.example.com");

}

}

activity\_main.xml: Toolbar를 레이아웃에 추가하여 화면 상단에 배치합니다.

옵션 메뉴 항목을 정의하는 XML 파일을 생성합니다. res/menu/menu\_main.xml

<item

android:id="@+id/action\_settings"

android:title="Settings"

android:orderInCategory="100"

android:showAsAction="never" />

메인 액티비티 (MainActivity.java) 설정

Toolbar를 설정하고 메뉴를 확장 및 처리하는 코드를 작성합니다.

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Toolbar를 설정합니다.

Toolbar toolbar = findViewById(R.id.toolbar);

setSupportActionBar(toolbar);

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// 메뉴를 확장합니다.

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.menu\_main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// 메뉴 항목 클릭을 처리합니다.

int id = item.getItemId();

if (id == R.id.action\_settings) {

Toast.makeText(this, "Settings clicked", Toast.LENGTH\_SHORT).show();

return true;

} else if (id == R.id.action\_about) {

Toast.makeText(this, "About clicked", Toast.LENGTH\_SHORT).show();

return true;

}

return super.onOptionsItemSelected(item);

}

}

옵션 메뉴와 달리 컨텍스트 메뉴는 원하는 위치에서 각각 호출

public class MainActivity extends AppCompatActivity {

private TextView textView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// TextView를 레이아웃에서 찾습니다.

textView = findViewById(R.id.textView);

// TextView에 컨텍스트 메뉴를 등록합니다.

registerForContextMenu(textView);

}

@Override

public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {

super.onCreateContextMenu(menu, v, menuInfo);

// 컨텍스트 메뉴의 헤더를 설정합니다.

menu.setHeaderTitle("Select Action");

// 컨텍스트 메뉴에 메뉴 아이템을 추가합니다.

getMenuInflater().inflate(R.menu.context\_menu, menu);

}

@Override

public boolean onContextItemSelected(MenuItem item) {

// 컨텍스트 메뉴의 항목을 클릭했을 때의 동작을 처리합니다.

switch (item.getItemId()) {

case R.id.action\_edit: // 수정 동작을 수행하는 코드

Toast.makeText(this, "Edit TextView", Toast.LENGTH\_SHORT).show();

return true;

case R.id.action\_delete: // 삭제 동작을 수행하는 코드

Toast.makeText(this, "Delete TextView", Toast.LENGTH\_SHORT).show();

return true;

default: return super.onContextItemSelected(item);

}

}

}