24장. 네트워크 정보 활용

24.1 다양한 네트워크 정보 활용

24.1.1. TelephonyManager

- 전화기로서의 기본 기능과 관련된 정보에 접근

PhoneStateListener
- 폰 상태 변경 감지

- onCallForwardingIndicatorChanged(boolean cfi): 통화 전달 상태 변화
- onCallStateChanged(int state, String incomingNumber): 통화 상태 변경
- onCellLocationChanged(CellLocation location): 스마트폰의 셀 위치 변경
- onDataActivity(int direction): 데이터 in/out 상태 변경
- onDataConnectionStateChanged (int state, int networkType): 데이터 연결 상태 변경
- onMessageWaitingIndicatorChanged(boolean mwi): 메시지 대기 상태 변경
- onServiceStateChanged(ServiceState serviceState): 단말기의 서비스 상태 변경
- onSignalStrengthsChanged(SignalStrength signalStrength): 신호 강도 변화

24.1 다양한 네트워크 정보 활용

```
PhoneStateListener listener=new PhoneStateListener(){
    @Override
    public void onServiceStateChanged(ServiceState serviceState) {
        super.onServiceStateChanged(serviceState);
    }
};
```

- PhoneStateListener을 상속받아 만든 클래스를 TelephonyManager에 등록하면 상태 변경을 감지

```
TelephonyManager telManager=(TelephonyManager)getSystemService(TELEPHONY_SERVICE);
telManager.listen(listener,PhoneStateListener.LISTEN_CALL_STATE);
```

- PhoneStateListener.LISTEN_CALL_FORWARDING_INDICATOR: 통화 전달 시
- PhoneStateListener.LISTEN_CALL_STATE: 통화 상태 변경 시
- PhoneStateListener.LISTEN_CELL_LOCATION: 기지국 변경 시
- PhoneStateListener.LISTEN_DATA_ACTIVITY: 데이터 송수신 활동 변경 시
- PhoneStateListener.LISTEN_DATA_CONNECTION_STATE: 데이터 연결 상태 변경 시
- PhoneStateListener.LISTEN_MESSAGE_WAITING_INDICATOR: 메시시 수신 대기 시
- PhoneStateListener.LISTEN_SERVICE_STATE : 전화기 상태 변경 시
- PhoneStateListener.LISTEN_SIGNAL_STRENGTHS : 전화 신호 세기 변경 시

24.1 다양한 네트워크 정보 활용

- 스마트폰에 전화가 걸려온 상태를 감지

```
public void onCallStateChanged(int state, String incomingNumber) {
    switch(state) {
        case TelephonyManager.CALL_STATE_IDLE:
            //...          break;
        case TelephonyManager.CALL_STATE_RINGING:
            //...
            break;
        case TelephonyManager.CALL_STATE_OFFHOOK:
            //...
            break;
    }
}
```

- TelephonyManager.CALL_STATE_IDLE: 통화 대기 상태로 변경
- TelephonyManager.CALL_STATE_RINGING: 전화벨이 울리는 상태로 변경
- TelephonyManager.CALL_STATE_OFFHOOK: 통화 중인 상태로 변경

24.1 다양한 네트워크 정보 활용

- 서비스 상태 변경을 감지

```
public void onServiceStateChanged(ServiceState serviceState) {
    switch(serviceState.getState()) {
        case ServiceState.STATE_EMERGENCY_ONLY:
            //...
            break;
        case ServiceState.STATE_IN_SERVICE:
            //...
            break;
        case ServiceState.STATE_OUT_OF_SERVICE:
            //...
            break;
        case ServiceState.STATE_POWER_OFF:
            //...
            break;
        default:
            //...
            break;
    }
}
```

- ServiceState.STATE_IN_SERVICE: 서비스 가능 상태
- ServiceState.STATE_EMERGENCY_ONLY: 긴급 통화만 가능한 상태
- ServiceState.STATE_OUT_OF_SERVICE: 서비스 불가 상태
- ServiceState.STATE_POWER_OFF: 비행모드 등 전화 기능을 꺼놓은 상태

24.1 다양한 네트워크 정보 활용

스마트폰 정보 획득
- getNetworkCountryIso(): 네트워크 제공 국가
- getNetworkOperatorName(): 네트워크 망 제공 사업자
- getNetworkType(): 네트워크 종류

```
datas.add("getNetworkCountryIso:"+telManager.getNetworkCountryIso());
datas.add("getNetworkOperatorName:"+telManager.getNetworkOperatorName());
if(telManager.getNetworkType()==TelephonyManager.NETWORK_TYPE_LTE){
    datas.add("getNetworkType : LTE");
}else if(telManager.getNetworkType()==TelephonyManager.NETWORK_TYPE_HSDPA){
    datas.add("getNetworkType : 3G");
}
```

- 국가코드 획득

```
Locale myLocale=getResources().getConfiguration().locale.getDefault();

Locale myLocale2=Locale.getDefault();
```

- 폰 번호 획득

```
telManager.getLine1Number()
```

24.1 다양한 네트워크 정보 활용

24.1.2. ConnectivityManager

네트워크 접속 정보를 확인

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

```
ConnectivityManager connManager = (ConnectivityManager)     getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connManager.getActiveNetworkInfo();
if (networkInfo != null) {
    if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI) {
        //...
    } else if (networkInfo.getType() == ConnectivityManager.TYPE_MOBILE) {
        //...    }
} else {
    //... 네트워크가 불가능한 상태 ...
}
```

- 브로드캐스트 리시버를 이용한 네트워크 상태 변경 감지

```
IntentFilter intentfilter = new IntentFilter();
intentfilter.addAction(ConnectivityManager.CONNECTIVITY_ACTION);
registerReceiver(networkReceiver, intentfilter);
```

- AndroidManifest.xml에 정적으로 등록해 실행하는 것이 금지
- 동적 등록에 의해 브로드캐스트 리시버를 실행하는 방법도 deprecated

24.1 다양한 네트워크 정보 활용

requestNetwork ( ), registerNetworkCallback ( ) 함수 이용 권장.
requestNetwork ( )는 필요한 순간 한 번 정보를 획득하는 함수이고, registerNetworkCallback ( )은 지속적으로 정보를 획득

함수	API 레벨
public void requestNetwork (NetworkRequest request, ConnectivityManager.NetworkCallback networkCallback)	21
public void requestNetwork (NetworkRequest request, ConnectivityManager.NetworkCallback networkCallback, int timeoutMs)	26
public void requestNetwork (NetworkRequest request, ConnectivityManager.NetworkCallback networkCallback, Handler handler)	26
public void requestNetwork (NetworkRequest request, PendingIntent operation)	22
public void requestNetwork (NetworkRequest request, ConnectivityManager.NetworkCallback networkCallback, Handler handler, int timeoutMs)	26

24.1 다양한 네트워크 정보 활용

- NetworkRequest에 addCapability ( )와 addTransportType ( ) 함수를 이용해 획득하고자 하는 네트워크의 정보를 설정하고, 결과는 콜백으로 받는다.

```
NetworkRequest networkReq = new NetworkRequest.Builder()
.addCapability(NetworkCapabilities.NET_CAPABILITY_INTERNET)
.addTransportType(NetworkCapabilities.TRANSPORT_CELLULAR)
.addTransportType(NetworkCapabilities.TRANSPORT_WIFI)
.build();
```

- NetworkRequest 객체를 requestNetwork( ) 함수에 NetworkCallback 객체와 같이 매개변수로 지정

```
connectivityManager.requestNetwork(networkReq, new ConnectivityManager.NetworkCallback() {
@Override
public void onAvailable(Network network) {
NetworkInfo networkInfo = connectivityManager.getNetworkInfo(network);
}
});
```

- registerNetworkCallback ( )

```
if (Build.VERSION.SDK_INT > Build.VERSION_CODES.LOLLIPOP) {
NetworkRequest.Builder builder = new NetworkRequest.Builder();
builder.addTransportType(NetworkCapabilities.TRANSPORT_WIFI);
connectivityManager.registerNetworkCallback(builder.build(), networkCallback);
}
```

24.1 다양한 네트워크 정보 활용

24.1.3. WifiManager

- 와이파이에 대한 접속 상태만을 판단

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
```

```
WifiManager wifiManager=(WifiManager)
        getApplicationContext().getSystemService(Context.WIFI_SERVICE);

if(!wifiManager.isWifiEnabled()){
    // 현재 wifi가 enable이 아니라면
    if(wifiManager.getWifiState() != WifiManager.WIFI_STATE_ENABLING){
        // wifi를 enable 상황으로 바꾼다
        wifiManager.setWifiEnabled(true);
    }
}
```

브로드캐스트 리시버를 이용한 변경 감지

- WifiManager.WIFI_STATE_CHANGED_ACTION : 와이파이 상태 변경
- WifiManager.NETWORK_STATE_CHANGED_ACTION : 와이파이 네트워크 연결 상태가 변경될 때
- WifiManager.RSSI_CHANGED_ACTION : 현재 연결된 와이파이 네트워크의 신호 세기가 변경될 때

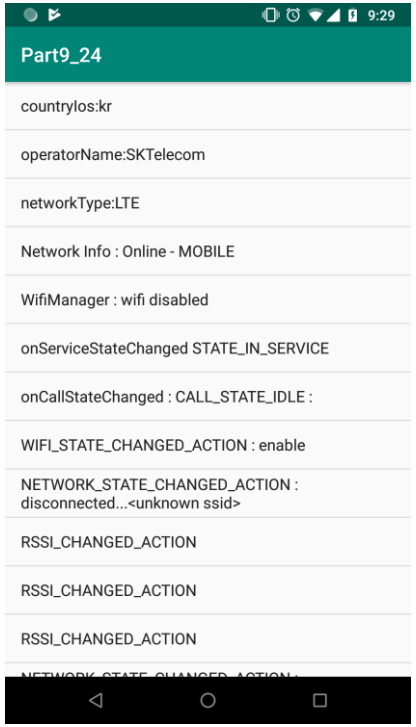
24.1 다양한 네트워크 정보 활용

```
IntentFilter wifiFilter=new IntentFilter();
wifiFilter.addAction(WifiManager.WIFI_STATE_CHANGED_ACTION);
wifiFilter.addAction(WifiManager.NETWORK_STATE_CHANGED_ACTION);
wifiFilter.addAction(WifiManager.RSSI_CHANGED_ACTION);
registerReceiver(wifiReceiver, wifiFilter);
```

```
BroadcastReceiver wifiReceiver=new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals(WifiManager.WIFI_STATE_CHANGED_ACTION)){
            int state = intent.getIntExtra(WifiManager.EXTRA_WIFI_STATE, -1);
            if (state == WifiManager.WIFI_STATE_ENABLED ) {
                //enable로 변경
            }
        }else if(intent.getAction()                    .equals(WifiManager.NETWORK_STATE_CHANGED_ACTION)){
            NetworkInfo networkInfo =                    intent.getParcelableExtra(WifiManager.EXTRA_NETWORK_INFO);
            WifiManager wifiManager =                    (WifiManager)context.getSystemService(Context.WIFI_SERVICE);
            WifiInfo wifiInfo = wifiManager.getConnectionInfo();
            String ssid = wifiInfo.getSSID();
            if (networkInfo.getState() == NetworkInfo.State.CONNECTED){
                //...          }else if (networkInfo.getState() == NetworkInfo.State.DISCONNECTED) {
                //...
            }
        }else if(intent.getAction().equals(WifiManager.RSSI_CHANGED_ACTION)){
            //...       }
    }
};
```

Step by Step 24-1 – 네트워크 정보 활용

- 모듈 생성
- 파일 복사
- AndroidManifest.xml 파일 작업
- MainActivity 작성
- 실행

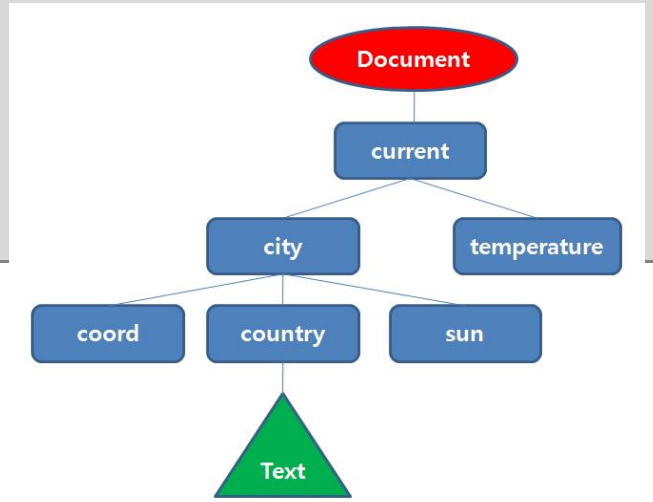


24.2 XML, JSON 데이터 파싱

24.2.1. XML 파싱

- DOM 파서
- XML 각 구성요소를 객체로 만들고 객체 간의 계층구조로 만들어 데이터를 추출하는 방식

```
<current>
   <city id="1835848" name="Seoul">
      <coord lon="126.98" lat="37.57"/>
      <country>KR</country>
      <sun rise="2017-03-28T21:21:49" set="2017-03-29T09:52:15"/>
   </city>
   <temperature value="11.01" min="7" max="13" unit="metric"/>
</current>
```



24.2 XML, JSON 데이터 파싱

- DOM 파서로 XML을 파싱하는 방법

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.parse(inputStream, null);
```

- Document DocumentBuilder.parse(InputStream stream [, String systemId])
- Document DocumentBuilder.parse(String uri)
- Document DocumentBuilder.parse(File file)

- XML 파싱으로 생성되는 객체는 Node 타입이며, 여러 Node의 집합 객체 타입이 NodeList

```
org.w3c.dom.Element root=doc.getDocumentElement();
NodeList nodeList=root.getChildNodes();
for(int i=0; i<nodeList.getLength(); i++){
    Node node=nodeList.item(i);
    if(node.getNodeType()==Node.ELEMENT_NODE){
        String tagName=node.getNodeName();
    }
    NodeList subList=node.getChildNodes();
        //...    }
}
```

24.2 XML, JSON 데이터 파싱

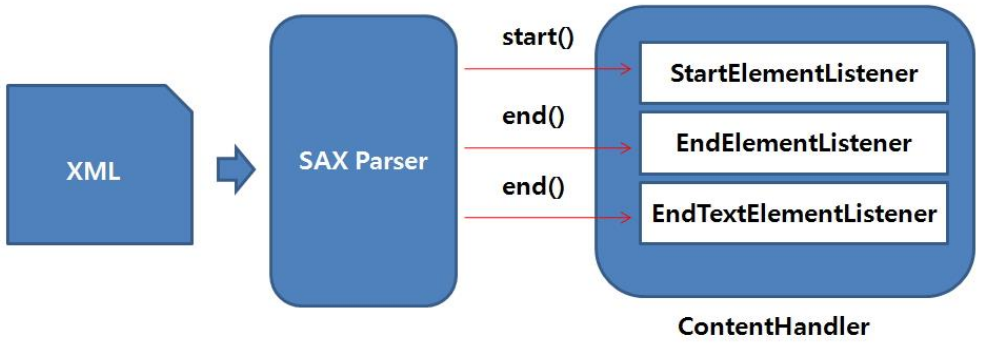
Node를 획득하기 위한 함수
- Node getFirstChild(): 하위 Node들 중 첫 번재 Node 획득
- Node getLastChild(): 하위 Node들 중 마지막 Node 획득
- Node getNextSibling(): 같은 계층의 Node 중 다음 순서의 Node 획득
- Node getPreviousSibling(): 같은 계층의 Node 중 이전 순서의 Node 획득
- Node getParentNode(): 상위 계층의 Node 획득
- NodeList getChildNodes(): 하위 Node의 집합 객체 획득

Node 객체를 Element 타입으로 변형하여 Element의 여러 함수로 태그와 관련된 다양한 데이터 획득
- getTagName(): 태그명 획득
- getAttribute(String name): 태그 속성값 획득
- getTextContent(): 태그의 보디 문자열 획득

```
org.w3c.dom.Element tempElement = (org.w3c.dom.Element) (doc.getElementsByTagName("temperature").item(0));
String temperature = tempElement.getAttribute("value");
```

24.2 XML, JSON 데이터 파싱

- SAX 파서
- SAX 파서는 객체를 만들지 않으며 단순히 이벤트만 발생시켜 등록된 이벤트 핸들러의 함수를 호출



- XML의 루트 태그를 RootElement 객체로 표현하며 그 하위의 파싱이 필요한 태그 객체를 Element 객체로 생성

```
RootElement root = new RootElement("current");
Element cityElement = root.getChild("city");
Element tempElement = root.getChild("temperature");
```

24.2 XML, JSON 데이터 파싱

- Element 객체에 이벤트 핸들러를 등록

```
cityElement.setStartElementListener(new StartElementListener() {
    @Override
    public void start(Attributes attributes) {
        cityView.setText(attributes.getValue("name"));
    }
});
```

- setElementListener(ElementListener elementListener)
- setEndElementListener(EndElementListener endElementListener)
- setEndTextElementListener(EndTextElementListener endTextElementListener)
- setStartElementListener(StartElementListener startElementListener)
- setTextElementListener(TextElementListener elementListener)

- RootElement를 SAX 파서인 XML 클래스의 parse( ) 함수에 전달하면 등록된 이벤트 핸들러의 함수가 호출

```
Xml.parse(inputStream, Xml.Encoding.UTF_8, root.getContentHandler());
```

24.2 XML, JSON 데이터 파싱

- Pull 파서

- Pull 파서도 SAX 파서와 마찬가지로 이벤트 중심의 파서
- Pull 파서는 특정 위치까지 파싱되어 내용을 처리한 후 계속 파싱할 것인지 멈출 것인지를 개발자가 제어할 수 있는 특징

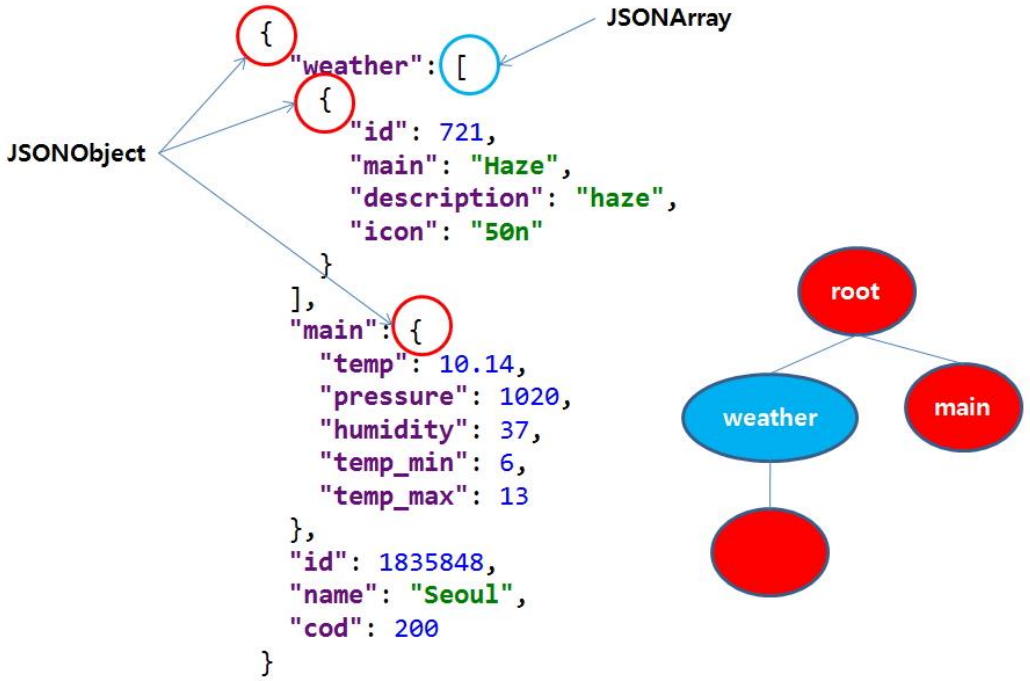
```
XmlPullParser parser = Xml.newPullParser();
parser.setInput(inputStream, null);
int eventType = parser.getEventType();
```

```
while (eventType != XmlPullParser.END_DOCUMENT) {
    String name = null;
    if (eventType == XmlPullParser.START_TAG) {
        name = parser.getName();
        if (name.equals("city")) {
            //...
        } else if (name.equals("temperature")) {
            //...
        }
    }
    eventType = parser.next();
}
```

24.2 XML, JSON 데이터 파싱

24.2.2. JSON 파싱

- JSON 파싱은 JSONObject 클래스와 JSONArray 클래스를 이용



24.2 XML, JSON 데이터 파싱

- JSONObject에 파싱할 JSON 문자열을 대입하면 최상위 JSONObject 객체가 만들어지며, 그 하위에 JSONObject 혹은 JSONArray 객체 여러 개가 만들어 진다.

```
JSONObject root = new JSONObject(json);
```

```
String name=root.getString("name");
JSONObject main=root.getJSONObject("main");
String temp=main.getString("temp");

JSONArray array=root.getJSONArray("weather");
JSONObject arrayItem=array.getJSONObject(0);
```

Step by Step 24-2 – 데이터 파싱

- 액티비티 생성
- 파일 복사
- Lab24_2Activity 작성
- 실행