Android

안드로이드 > Webview - WebViewClient의 메서드

먼저 WebView의 만드는 방법을 알아 봤다.

근데 WebView를 만든뒤에 확인해 보면 WebView에서 새로운 Url 접속이 일어 날면

새로운 창이 뜨는걸 볼 수 있다. 이는 기본적으로 WebView에서 새로운 Url 접속시

시스템에서 새로운 창에다가 로딩하도록 시키는 것인데, 이것이 한두번씩 많아지면

메모리 관리상에도 별로 도움이 안된다. 그럼 어찌해야 될까..

그래서 있는 것이 WebViewClient 와 WebChromeClient 이다.

이번 장에는 webViewClient class 에 대해서 알아보자.

일단 WebViewClient 는 새로운 Class에 WebViewClient를 상속 받아서 만들어도 되고

```
private class WebClientClass extends WebViewClient {
      ~~~~~~ 내용 ~~~~~~~~~~~~~~~~~~~~~~~~
}
```

내부에서 new로 생성해도 된다.

```
WebView.setWebViewClient(new WebViewClient(){
      ~~~~~~ 내용 ~~~~~~~~~~~~~~~~~~~~~~~~
}
```

WebViewClient Class 함수들에 대해서 알아보자.

1. onPageStarted()

로딩이 시작될때..

WebView에서 처음 한 번만 호출되는 메쏘드 페이지 로딩이 시작된 것을 알립니다.

```
@Override
public void onPageStarted(WebView view, String url, Bitmap favicon) {
   super.onPageStarted(view, url, favicon);
}
```

2. onLoadResource()

WebView가 주어진 URL로 지정된 리소스를 로드할 것이라고 알립니다.

페이지 로딩이 완료될 때까지 여러번 호출됩니다. 페이지가 나뉘어서 로딩되나 봅니다.

```
@Override
public void onLoadResource(WebView view, String url) {
	super.onLoadResource(view, url);
}
```

3. doUpdateVisitedHistory()

방문한 링크를 데이터베이스에 업데이트한다고 알립니다.

```
@Override
public void doUpdateVisitedHistory(WebView view, String url, boolean isReload) {
	Log.i("WebView", "History: " + url );
	super.doUpdateVisitedHistory(view, url, isReload);
}
```

```
/* 결과 (url이 변할 때 마다)
History: http://www.abc.com/djfk...
History: http://www.abc.com/djfk.../fdfd
*/
```

4. onPageFinished()

WebView에서 처음 한 번만 호출되는 메쏘드. 페이지 로딩이 완료된 것을 알립니다.

```
@Override
public void onPageFinished(WebView view, String url) {
	super.onPageFinished(view, url);
}
```

호스트 응용 프로그램에게 오류를 보고합니다. 이러한 오류는 복구할 수 없습니다.

웹뷰는 인터넷이 열결되어 있지 않았을때 주소가 노출되는 단점이 있다. 이럴경우 url주소를 보안상 노출되면 안되기 때문에 숨길경우 사용하면 유용할 것 같

```
@Override
public void onReceivedError(WebView view, int errorCode,String description, String failingUrl) {
	super.onReceivedError(view, errorCode, description, failingUrl);

	switch(errorCode) {
		case ERROR_AUTHENTICATION: break;              // 서버에서 사용자 인증 실패
		case ERROR_BAD_URL: break;                     // 잘못된 URL
		case ERROR_CONNECT: break;                     // 서버로 연결 실패
		case ERROR_FAILED_SSL_HANDSHAKE: break;    // SSL handshake 수행 실패
		case ERROR_FILE: break;                        // 일반 파일 오류
		case ERROR_FILE_NOT_FOUND: break;              // 파일을 찾을 수 없습니다
		case ERROR_HOST_LOOKUP: break;          // 서버 또는 프록시 호스트 이름 조회 실패
		case ERROR_IO: break;                          // 서버에서 읽거나 서버로 쓰기 실패
		case ERROR_PROXY_AUTHENTICATION: break;   // 프록시에서 사용자 인증 실패
		case ERROR_REDIRECT_LOOP: break;               // 너무 많은 리디렉션
		case ERROR_TIMEOUT: break;                     // 연결 시간 초과
		case ERROR_TOO_MANY_REQUESTS: break;     // 페이지 로드중 너무 많은 요청 발생
		case ERROR_UNKNOWN: break;                     // 일반 오류
		case ERROR_UNSUPPORTED_AUTH_SCHEME: break; // 지원되지 않는 인증 체계
		case ERROR_UNSUPPORTED_SCHEME: break;          // URI가 지원되지 않는 방식
	}
}
```

6. onReceivedHttpAuthRequest()

* 인증 요청을 처리한다고 알립니다. 기본 동작은 요청을 취소하는 것입니다. (http 인증요청이 있을 경우)

```
@Override
public void onReceivedHttpAuthRequest(WebView view,HttpAuthHandler handler, String host, String realm) {
	super.onReceivedHttpAuthRequest(view, handler, host, realm);
}
```

6. onScaleChanged()

스케일이 변경되었을 때 처리할 내용을 구현한다. (확대나 크기등이 변화 있을 경우)

```
public void onScaleChanged(WebView view, float oldScale, float newScale) {
        super.onScaleChanged(view, oldScale, newScale);
}
```

7. onUnhandledKeyEvent()

잘못된 키 입력이 있을 경우 호출되는 메쏘드

시스템 키를 제외하고, WebView는 shouldOverrideKeyEvent가 true를 반환하는 경우나

일반적인 flow에서 항상 키 이벤트를 처리합니다. 키 이벤트가 발생된 곳으로 부터

비동기적으로 호출됩니다.

```
@Override
public void onUnhandledKeyEvent(WebView view, KeyEvent event) {
        super.onUnhandledKeyEvent(view, event);
}
```

8. shouldOverrideKeyEvent()

사용자의 키 입력이 있을 경우 호출되는 메쏘드

시스템 키를 제외하고, WebView는 shouldOverrideKeyEvent가 true를 반환하는 경우나

일반적인 flow에서 항상 키 이벤트를 처리합니다. 키 이벤트가 발생된 곳으로 부터

비동기적으로 호출됩니다.

```
@Override
public void shouldOverrideKeyEvent(WebView view, KeyEvent event) {
        return super.shouldOverrideKeyEvent(view, event);
}
```

(예제) shouldOverrideKeyEvent()

```
@Override
public boolean shouldOverrideKeyEvent(WebView view, KeyEvent event) {
        int keyCode = event.getKeyCode();
        Log.d("HelloWebViewClient","#### shouldOverrideKeyEvent()  " + keyCode);
```

```
            webview.goBack();    //취소버튼시 전 페이지로 간다.
            return true;
        }else if ((keyCode == KeyEvent.KEYCODE_DPAD_RIGHT) && webview.canGoForward()) {
            webview.goForward();        //오른쪽 버튼시 앞 페이지로 간다.
            return true;
        }

        return false;
    }
```

9. shouldOverrideUrlLoading()

새로운 URL이 현재 WebView에 로드되려고 할 때 호스트 응용 프로그램에게 컨트롤을 대신할 기회를 줍니다

```
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        //return super.shouldOverrideUrlLoading(view, url);
        view.loadUrl(url);
        return true;
    }
```