Table of Contents

estgames-common-framework (이스트 게임즈 안드로이드 공통 라이브러리)

이스트 게임즈 안드로이드 공통 라이브러리 관련 메뉴얼입니다.

해당 라이브러리는 공통적인 기능과 유저계정 관련한 기능을 담고 있습니다.

공통적인 기능은 배너, 이용약관, 권한, 공지사항, 고객센터, 이벤트, 나라, 언어, 로컬 관련 기능을 처리합니다.

유저 계정은 게스트 로그인, Sns로그인 관련한 기능을 처리합니다.

보시고 잘 모르시겠는 거는 웹플랫폼팀에 문의 주시면 됩니다.

EG 모바일 플랫폼 SDK for Android 튜토리얼

이 문서는 안드로이드 모바일 APP에 EG 플랫폼 설정에서 계정 연동까지를 간단하고 빠르게 적용하는 하는 방법을 설명하고 있습니다.

튜토리얼은 세가지 단계로 진행됩니다.

1. App 설정하기
2. 플랫폼 기본 스타트 액션 호출하기
3. 플랫폼 세션 연동하기

App 설정하기

이 튜토리얼 문서는 EG 플랫폼을 빠르게 연동하기 위한 기본적인 설정 방법을 설명하고 있습니다. 좀 더 자세한 내용은 라이브러리 설정 문서를 참조하기 바랍니다. 이 튜토리얼 문서는 안드로이드 프로젝트가 Gradle로 구성되어 있다고 가정하고 진행합니다. 다른 빌드 환경을 사용하는 경우는 해당 환경에 맞춰서 의존성 트리를 구성해주세요.

bulid.gradle 설정

1. EG 플랫폼 SDK for Android 라이브러리 등록.
2. 다운로드 받은 `mp-aos-release.aar` 파일을 프로젝트 라이브러리 디렉토리(`libs`)에 복사합니다.
3. `gradle` 에서 라이브러리를 감지 할 수 있도록 `repositories` 설정을 추가합니다.

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

1. 의존 라이브러리 설정
2. 프로젝트가 필요로 하는 라이브러리를 등록합니다.

```
dependencies {
    implementation 'org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.+'
    implementation 'com.android.support:multidex:1.0.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'
    implementation 'com.google.android.gms:play-services-auth:15.0.1'
    implementation 'com.facebook.android:facebook-login:[4, 5)'
    implementation 'com.estgames.framework:mp-aos-sdk-release:2.0@aar'
}
```

raw 디렉토리에 플랫폼 설정정보 등록

EG 플랫폼 연동을 위해 등록한 Application 연동 설정 파일을(`egconfigration.json`) 프로젝트의 `res/raw` 디렉토리에 저장합니다.

```
# res/raw/egconfiguration.json

{
  "Client" : {
    "ClientId": "0eae170b-ec59-3f85-bfda-a8c80fc1a3fe.mp.estgames.com",
    "Secret": "13949b24f1fd9d3c81aabd11295c28507ee0a977e9dc5dc93a3bea86f8243b46",
    "Region": "cm.global.stage"
  },

  "Account": {
    "Google": {
      "Scopes": "email, profile, openid"
    },
    "Facebook": {
      "Scopes": "public_profile, email"
    }
  }
}
```

Application 클래스 작성

App 전역에서 플랫폼 Context 참조를 위한 Application 클래스를 작성합니다. 작성 위치는 App의 root package 입니다.

```
public class Application extends MultiDexApplication implements PlatformContextContainer {
    private PlatformContext delegateContext;

    @Override
    public PlatformContext getContext() {
        return delegateContext;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        this.delegateContext = new GenericPlatformContext(getApplicationContext());
    }
}
```

AndroidManifest.xml 파일 작성

1. package name 확인
2. SNS 연동을 하는 경우 SNS 프로바이더의 App 또는 Application 에 등록한 패키지 명이 맞지 않을 경우 로그인 인증이 거부 됩니다. 등록한 패키지 명이 맞는지 반드시 확인하세요.
3. 프로젝트 구성 패키지와 외부 노출 패키지가 다른경우 `gradle` 설정에서 외부 노출 패키지명을 설정 할 수 있습니다.

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    pacakage="net.sample.android.app">
```

1. uses-permission 설정

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

1. application context 클래스 등록
2. 위에서 플랫폼 참조를 위해 작성한 Application 클래스를 `AndroidManifest.xml` 파일에 등록합니다.

```
<application android:name=".Application" ...>
```

플랫폼 기본 스타트 액션 호출하기

이 튜토리얼 문서는 EG 플랫폼을 연동하고 간단하게 플랫폼 시작 액션을 호출 하는 방법을 설명하고 있습니다. 좀 더 자세한 내용은 EgClient 메뉴얼를 참조하기 바랍니다. 이 튜토리얼 문서는 사용자의 명시적인 로그인을 통해 세션을 생성한다고 가정하고 진행합니다.

`EgClient` 생성

`EgClient` 객체는 EG 플랫폼의 Context 정보 및 기본 액션들을 포함 하고 있는 객체입니다. `EgClient` 객체는 App 내에서 전역적으로 한개의 객체만 생성되고 유지됩니다.

```
EgClient client = EgClient.from(getApplication());
//or
EgClient client = EgClient.from(getApplicationContext();
```

Starting Action

EG 플랫폼을 연동하는 경우 플랫폼에서 제공하는 액션들을(약관, 배너, 공지사항 등) 사용할 수 있습니다. 시작 액션은 앱이 시작할때 동작해야 하는 액션 프로세스를 정의한 액션입니다. Starting Action을 호출하는 것으로 앱 시작과 동시에 진행 되어야 할 액션 프로세스를 진행 할 수 있습니다.

```
    // Starting Action 객체 생성
    Action starting = client.starging();

    // Starting Action Listener 적용
    starting.setListener(new ActionListener<String>() {
         @Override
        public void onDone(String response) {
            // 프로세스 완료 핸들러 작성
        }

        /** 생략가능 **/
        @Override
        public void onCancel() {
            // 프로세스 취소 핸들러 작성
        }

        /** 생략가능 **/
        @Override
        public void onError(Throwable t) {
            // 프로세스 에러 핸들러 작성
        });

    // Action 실행
    starting.go();
```

플랫폼 세션 연동하기

이 튜토리얼 문서는 EG 플랫폼과 빠르게 계정연동을 하기 위한 기본적인 설정 방법을 설명하고 있습니다. 좀 더 자세한 내용은 플랫폼 계정 연동 문서를 참조하기 바랍니다. 이 튜토리얼 문서는 사용자의 명시적인 로그인을 통해 세션을 생성한다고 가정하고 진행합니다.

세션 매니저 생성

`SessoinManager` 는 EG 플랫폼의 세션 제어를 담당하는 클래스 입니다. 세션을 생성하고 갱신, 조회, 삭제 하기 위해서는 세션매니저를 생성해야 합니다. `SessionManager` 는 EgClient 를 통해 생성할 수 있습니다.

```
    SessionManager sessionManager = Client.from(context).getSessionManager();
```

Guest 세션 생성하기 (Guest 사용자 로그인하기)

EG 플랫폼에서 Guest 사용자는 SNS 인증을 거치지 않은 비로그인 세션입니다. Guest 세션은 SNS 사용자와 연동을 거치지 않기 때문에 세션 만들기로 바로 생성할 수 있습니다.

주의 : 이전에 생성된 세션이 있는경우 호출하면 세션 중복생성 에러를 던지게 됩니다.

```
if (!sessionManager.isSessionOpen) {
    sessionManager.crate()
            .onError(new Task.Acceptor<Throwable>() {
                @Override
                public void accept(Throwable t) {
                    // 세션 생성 실패 핸들러 작성
                }
            })
            .asyncAccept(new Task.Acceptor<String>() {
                @Override
                public void accept(String token) {
                    // 세션 생성 성공 핸들러 작성
                }
            });
}
```

세션 토큰 조회

`SessionManager` 를 통해 현재 세션의 토큰 정보를 조회 할 수 있습니다. 세션이 열려있지 않은경우 `null` 을 반환합니다.

```
    Token token = sessionManager.getToken();

    String egToken = token.getEgToken();
    String refToken = token.getRefreshToken();
```

사용자 Profile 조회

`SessionManager` 를 통해 현새 세션의 사용자 Profile 정보를 조회 할 수 있습니다. 세션이 열려있지 않은경우 `null` 을 반환합니다.

```
    Profile profile = sessionManager.getProfile();

    String egId = profile.getEgId();
```

```
    String userId = profile.getUserId();
    String email = profile.getEmail();
    String provider = profile.getProvider();
```

세션 재시작하기

EG 플랫폼의 세션 토큰은 만료시간을 가지고 있습니다. 따라서 앱을 다시 시작했을 경우 유효한 세션을 만들기 위해 앱을 시작할때 마다 세션을 재시작 해야 합니다.

```
if (sessionManager.isSessionOpen) {
    sessionManager.resume()
            .onError(new Task.Acceptor<Throwable>() {
                @Override
                public void accept(Throwable t) {
                    // 세션 갱신 실패 핸들러 작성
                }
            })
            .asyncAccept(new Task.Acceptor<String>() {
                @Override
                public void accept(String token) {
                    // 세션 갱신 성공 핸들러 작성
                }
            });
}
```

세션 종료(로그아웃)하기

`SessionManager` 를 이용하여 현재 발행된 세션을 종료 할 수 있습니다. 세션을 종료하면 토큰을 만료하고 사용자 연결을 끊습니다. (SNS 로그인이 되어 있다면 SNS 연결도 로그아웃 시킵니다.)

```
    sessionManager.signOut()
            .onError(new Task.Acceptor(Throwable t) {
                // 세션 종료 실패 핸들러 작성
            })
            .asyncAccept(new Task.Acceptor() {
                // 세션 종료 핸들러 작성
            })
```

SNS(외부 프로바이더) 계정 연동하기

EG 플랫폼 계정은 SNS(외부 프로바이더) 계정과 연동 및 인증이 가능합니다. SNS 계정을 연동하게 되면 Guest 계정과는 달리 로그인을 통하여 사용자의 계정정보를 유지 할 수 있습니다.

이 문서는 플랫폼 SDK에서 제공하는 로그인 대화상자를 이용해 연동하는 방법을 안내합니다. 플랫폼 SDK는 로그인 대화상자를 비롯한 계정연동을 위한 다른 여러 옵션들을 제공하고 있습니다. 더 자세한 정보는 SNS 계정연동 문서를 참고하시기 바랍니다.

1. `SignInControl` 객체생성

```
    SignInControl.Option option = new SignInControl.Option()
                .setSignInResultHandler(new SignInResultHandler() {
                    @Override
                    public void onComplete(Result.Login result) {
                        // 로그인 성공 핸들러 작성
                    }

                    @Override
                    public void onError(Throwable t) {
```

```
                        // 로그인 실패 핸들러 작성
                    }

                    @Override
                    public void onCancel() {
                        // 로그인 취소 핸들러 작성
                    }
                });

    SignInControl signInControl = SignInControl.createControl(activity, option);
```

1. `onActivityResult` 메소드 작성

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    signInControl.onActivityResult(requestCode, resultCode, data);
}
```

1. 로그인 화면 호출

```
signInControl.signIn(activity);
```

라이브러리 설정

해당 라이브러리를 사용하기 앞서 몇가지 초기설정이 필요합니다. 해당 장에서 순서대로 필요한 설정을 설명합니다.

- build.gradle 파일 dependency 설정
- AndroidManifest.xml 작성
- Client 설정
- PlatformContext 구성

gradle 파일 의존성 설정

안드로이드 프로젝트는 기본적으로 gradle을 기반으로 프로젝트를 구성합니다. EGMP SDK를 사용하기 위해 build.gradle 파일에 의존성 정보를 등록합니다.

- 해당 라이브러리는 코틀린로 작성되었습니다. 코틀린 런타임 라이브러리를 적용해주세요.
  - 'org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.+'
- Appication 클래스 작성에 필요한 라이브러리
  - 'com.android.support:multidex:1.0.1'
  - 'com.android.support.constraint:constraint-layout:1.1.2'
- SNS 계정연동에 사용되는 라이브러리
  - 'com.google.android.gms:play-services-auth:15.0.1'
  - 'com.facebook.android:facebook-login:[4, 5)'
- EstGame 라이브러리(제공되는 aar 파일을 app/libs 폴더에 넣어주세요.)
  - implementation 'com.estgames.framework:mp-aos-sdk-release:1.0@aar'
- defaultConfig에 applicationId에 있는 패키지명을 해당게임에 맞는 패키지명을 일치 시켜주어야 구글 로그인이 가능합니다. 자세한 패키지명은 웹플랫폼팀에 문의 부탁드립니다.

build.gradle

```
defaultConfig {
        applicationId "$app_package_name"
        .
        .
        .
    }

dependencies {
    .
    .
    .

    implementation 'org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.+'

    implementation 'com.android.support:multidex:1.0.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.2'

    implementation 'com.google.android.gms:play-services-auth:15.0.1'
    implementation 'com.facebook.android:facebook-login:[4, 5)'

    implementation 'com.estgames.framework:mp-aos-sdk-release:2.0@aar'
}

//app/libs 폴더에 넣은 aar파일을 읽기 위한 설정
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

AndroidManifest.xml 설정

안드로이드 앱에서 MP SDK 의 라이브러리를 사용하기 위해서는 안드로이드의 Manifest.xml 파일에 몇가지 설정이 추가 되어야 합니다.

package name 설정

사용자 계정을 외부 OPEN ID 서비스의 계정을 사용하여 연동할 경우 각 프로바이더들의 클라이언트 설정에 package name 을 필요로 합니다. IDE 툴을 사용하여 프로젝트를 생성했을 경우 package name 을 수정할 필요는 없습니다.

> 일부 OPEN ID 서비스의 경우 (google+) package name 정보가 일치하지 않는 경우 계정 연동에 실패합니다.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" pacakage="net.sample.android.app">
    ...
</manifest>
```

uses-permission 설정

EGMP 서비스들은 모두 웹 기반의 API 입니다. 따라서 모바일 앱에서 인터넷 및 네트워크 관련 권한설정을 해야 합니다.

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

Facebook 로그인 설정 (Option)

Android App 에서 Facebook 계정으로 로그인 연동을 하는 경우 Manefest.xml 파일에 페이스북 관련 설정을 해야합니다. 페이스북 로그인은 페이스북 로그인 창을 위한 Activity 등록과 페이스북 앱 인증을 위한 meta-data 설정이 필요합니다. facebook app id 등의 리소스를 분리할 경우 res/values/string.xml 파일에 리소스를 등록해줍니다.

Facebook Login Activity 등록

```
<manifest ... >
    <application ...>
        <activity
            android:name="com.facebook.FacebookActivity" android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="@string/fb_login_protocol_scheme" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Facebook meta-data 설정

```
<manifest ... >
    <application ...>
```

```
        <meta-data
            android:name="com.facebook.sdk.ApplicationId"
            android:value="@string/facebook_app_id" />
    </application>
</manifest>
```

Facebook 설정을 위한 리소스 등록 (res/values/string.xml)

fb_login_protocol_scheme 값은 fb + [facebook_app_id] 형태로 입력해주시면 됩니다.

> 예) fb1234567893456

```
<resource>
    <string name="facebook_app_id">[facebook_app_id]</string>
    <string name="fb_login_protocol_scheme">fb[facebook_app_id]</string>
</resource>
```

Client 설정정보 import

EGMP SDK 는 res/raw 디렉토리에 위치한 설정 파일을 읽어 framework 초기화를 실행합니다. 따라서 SDK에서 필요한 설정 파일을 해당 디렉토리(res/raw) 아래에 위치시켜야 합니다. 필요한 파일은 다음과 같습니다.

- egconfiguration.json

설정 파일은 MP Console 에서 다운로드를 제공할 예정입니다.

```
{
  "Client" : {
    "ClientId": "sample-client-id.sample.estgames.com",
    "Secret": "sample-app-secret-code",
    "Region": "sample.global.stage"
  },

  "Account": {
    "Guest": {
      "Identity": "Device"
    },
    "Google": {
      "Scopes": "email, profile, openid"
    },
    "Facebook": {
      "Scopes": "public_profile, email"
    }
  }
}
```

Client

앱의 인증 및 기본 설정 정보를 지정합니다. 클라이언트 아이디, 앱 시크릿, Region 정보를 설정합니다.

- ClientId : 클라이언트 아이디
- Secret : 앱 시크릿
- Region : 서비스 Region 코드

Account

앱의 사용자 계정에 대한 설정 정보를 지정합니다. 게스트의 식별자 생성방법, SNS 프로바이더등을 설정합니다.

- Guest : 게스트 계정에 대한 설정을 합니다.
  - Identity - 게스트 계정 식별자 생성 방법을 설정합니다. 설정값은 device, app, instant 중 하나를 설정하면 됩니다.
- Google, Facebook : 구글, 페이스북 연동 설정을 합니다.
  - Scopes - SNS 계정 연동시 요청할 퍼미션 정보를 설정합니다.

PlatformContext 구성

EGMP 서비스를 이용하기 위해 MP Configuration을 초기화 하고 Context를 구성하도록 합니다. Application 클래스를 MP Context로 인식 할 수 있도록 PlatformContextContainer 인터페이스를 구현합니다.

Application 클래스 추가

Application.java

```
public class Application extends MultiDexApplication implements PlatformContextContainer {
    private PlatformContext delegateContext;

    @Override
    public PlatformContext getContext() {
        return delegateContext;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        this.delegateContext = new GenericPlatformContext(getApplicationContext());
    }
}
```

Application 클래스 등록

초기화 코드를 담고 있는 Appication 를 앱이 인식 할 수 있도록 AndroidManifest.xml 파일에 등록해줍니다.

AndroidManifest.xml

```
<manifest ... >
    <application
        android:name='.Application'
        ... >
    ...
    </application>
</manifest>
```

Configuration 설정하기

`ConfigurationScript` 인터페이스를 이용해 앱 설정값을 작성 할 수 있습니다. 이 인터페이스를 이용하면 `egconfiguration.json` 설정 파일없이 순수하게 자바 코드만으로 어플리케이션 설정을 구성할 수 있습니다.

```
    class SampleScript implements ConfigurationScript {
        @Override
        public void script(Configuration config) {
            config
                .app()
                    .region("sample.region.code")
                    .clientId("sample-client-id")
                    .secret("sample-app-secret")
                    .end()
                .account()
                    .guest().identityGenerator(DeviceUserIdentityGenerator.class)
```

```
                    .facebook().scopes("public_profile, email");
        }
    }
```

> configuraion 을 구성하는 각 섹션 설정 스크립트는 `end()` 메소드로 종료할 수 있습니다.

App (client) 설정

`Configuration` 객체의 `app()` Descriptor 를 이용하면 클라이언트 설정을 작성 할 수 있습니다. App 설정은 `ClientID`, `Secret`, `Region` 으로 구성됩니다.

```
public void script(Configuration config) {
    config
        .app()
            .region("region-code")              // Region 코드 설정
            .clientId("app-client-id-value")    // client id 설정
            .secret("app-secret-value");        // app secret 설정
}
```

Account 설정

`Configuration` 객체의 `account()` Descriptor 를 이용하면 사용자 계정 설정을 작성 할 수 있습니다. Account 설정은 게스트, SNS 프로바이더 설정으로 구성됩니다.

```
public void script(Configuration config) {
    config
        .account()
            .guest().identityGenerator(InstantUserIdentityGenerator.class)
            .google().scopes("email, profile, openid")
            .facebook().scopes("public_profile", "email");
}
```

- `guest()` - 게스트 계정관련 설정을 작성합니다.
  - identityGenerator : 게스트 계정 식별자 생성기를 설정합니다. 기본값은 DeviceUserIdentityGenerator.class 입니다.
- `google()`, `facebook()` - SNS 계정 프로바이더 설정을 작성합니다.
  - scopes : SNS 계정 연동시 요청할 퍼미션 정보를 설정합니다.

Java Code Configuration 예제

다음은 ConfigurationScript 인터페이스를 이용해 어플리케이션 설정을 작성하는 예제 `Application` 클래스입니다.

```
class Application extends MultiDexApplication implements PlatformContextContainer, ConfigurationScript {
    private GenericPlatformContext delegateContext;

    @Override
    public void onCreate() {
        super.onCreate();
        delegateContext = new GenericPlatformContext(getApplicationContext());
        // Configuration Script를 PlatformInitializer에 적용합니다.
        // GenericPlatformContext 클래스는 PlatformInitializer 인터페이스를 구현하고 있습니다.
        // PlatformContext 객체는 EgClient 를 통해 Context 를 얻어올때 초기화가 됩니다.
        delegateContext.configuration(this);
    }
```

```
    @NotNull @Override
    public PlatformContext getContext() {
        return delegateContext;
    }

    /**
     * Configuration Script 를 작성합니다.
     **/
    @Override
    public void script(Configuration config) {
        config
            .app()
                .region("sample.global.stage")
                .clientId("sample-client-id.sample.estgames.com")
                .secret("sample-app-secret-code")
                .end()
            .account()
                .guest().identityGenerator(DeviceUserIdentityGenerator.class)
                .facebook).scopes("public_profile, email")
                .google().scopes("email", "profile", "openid");

    }
}
```

EgClient 메뉴얼

이용약관, 배너, 공지... 등 게임에서 사용되는 공통 모듈은 `EgClient` 클래스에 정의되어있습니다.

`EgClient` 객체에서 각각의 팝업창으로 나오는 기능에 대해서는 `Action` 객체를 생성처리하며 나라, 국가관련 정보는 문자열로 리턴합니다.

`Action` 객체에서는 팝업창 관련 클래스입니다. 해당 객체에서는 일어나야할 동작을 등록하고 기능을 실행합니다.

`ActionListener` 인터페이스에서는 팝업창에서 성공, 취소, 에러시 일어나야 할 동작을 구현합니다.

이 장은 아래와 같은 내용으로 구성됩니다.

- EgClient 생성합니다.
- 각각의 팝업에 대한 Action을 생성하고 실행합니다.
- ActionListener 구현 하고 Action에 등록합니다.
- EgClient 그외 함수들의 대해 알아봅니다.

객체 생성

`EgClient` 에서는 원하는 기능들에 대한 `Action` 들을 생성합니다. 여기서는 `EgClient` 를 초기화하는 함수를 알아봅니다.

객체 생성 함수

메소드 이름	리턴 값	매개변수
from	com.estgames.framework.EgClient	android.content.Context

예)

```
EgClient client = EgClient.from(getApplication());
//or
EgClient client = EgClient.from(getApplicationContext();
```

Action

`Action` 은 해당 라이브러리에서 제공하는 팝업창 동작에 대한 클래스입니다. 앞에서 생성한 `EgClient` 에서 원하는 팝업창에 대한 `Action` 을 생성합니다.

EgClient에서 Action 클래스를 만드는 함수

- 해당 함수들은 공통적으로 `com.estgames.framework.Action` 클래스를 리턴한다.
- `android.app.Activity` 을 매개변수로 받는 다. 해당 팝업을 띄울 액티비티를 매개변수로 받는 다.
- `Action` 클래스에 `go` 함수로 팝업창을 띄울 수 있다.

메소드 이름	설명
starting	API에서 지정된 순서대로 팝업창을 띄우는 액션
authority	권한
banner	배너
policy	이용약관
notice	공지사항
cs	고객센터
event	이벤트

예)

```
Action<String> starting = client.starting(this);
starting.go();
```

ActionListener 구현 및 등록

`ActionListener` 에서는 팝업창에서 정상종료, 취소, 에러시 일어나야할 작업을 구현한다.

이름	설명	매개변수
onDone	정상종료	성공 결과값(타입은 제너릭), 팝업마다 결과타입이 다르다.
onCancel	X버튼 혹은 취소	
onError	에러	에러핸들러로 에러종류를 구분합니다.

예)

```
Action<String> starting = client.starting(this);

starting.setListener(new ActionListener<String>() {
    @Override
    public void onDone(String response) {
        // 프로세스 완료 핸들러 작성
    }

    /** 생략가능 **/
    @Override
    public void onCancel() {
        // 프로세스 취소 핸들러 작성
    }

    /** 생략가능 **/
    @Override
    public void onError(Throwable t) {
        // 프로세스 에러 핸들러 작성
    }
});

starting.go();
```

EgClient 클래스의 그외 함수

함수명	리턴타입	설명
setLang	void	라이브러리 언어를 설정한다. 한글, 영어 지원
getLang	String	설정된 언어가 없을 경우 기기에 설정된 언어가 리턴된다.
getLocale	java.util.Locale	로케일 정보를 가져온다.
getNation	String	국가 정보 리턴
getSessionManager	SessionManager	세션매니저 생성

플랫폼 계정 연동

플랫폼 계정 연동에 가장 먼저할 작업은 `SessionManager` 를 생성하는 일입니다. 매니져에서는 게스트 유저로의 로그인, 유저정보(세션) 조회, 로그아웃(세션종료)을 처리합니다.

`SessionManager` 는 토큰 생성, 갱신, 만료, 세션 종료등 EG 모바일 플랫폼의 세션 및 토큰 관리를 담당하는 객체입니다.

Sns 연동은 `SignInControl` 에서 처리하며 라이브러리에서 제공하는 버튼을 사용하여 화면을 만들거나 기본으로 제공하는 로그인 팝업창을 사용하여 로그인을 처리합니다.

이 장은 아래내용으로 구성됩니다.

게스트 계정의 식별자 생성

EG 플랫폼 SDK 에서는 3가지 유형으로 게스트 계정의 식별자를 생성 할 수 있습니다. 각 유형에 따라 기기에서 게스트 계정을 지속 가능한 또는 휘발성 계정으로 생성 할 수 있습니다.

게스트 계정의 생성 유형은 `confgiruation` 또는 *egconfiguration.json* 파일에서 설정 할 수 있습니다.

Device 기반 식별자 생성

디바이스 기반으로 게스트 계정 식별자를 생성하게 되면 한 디바이스당 하나의 게스트 계정만 생성합니다. 앱을 삭제한 후 재설치를 하는 경우에도 같은 게스트 계정의 유지를 보장합니다.

> 기기에 따라서 앱 삭제 후 재설치시 같은 게스트 계정을 보장하지 못 할 수도 있습니다.

EG 플랫폼에서는 이 모드를 기본값으로 설정합니다.

- 설정 옵션
  - `configuration` : DeviceUserIdentityGenerator.class
  - json 설정 파일 : device

App 기반 식별자 생성

앱 기반으로 게스트 계정 식별자를 생성하게 되면 같은 디바이스에서 앱 별로 게스트 계정 유지를 보장합니다. 앱을 삭제한 후 재설치를 하는 경우에는 게스트 계정을 유지하지 않습니다. 즉 앱을 재설치하는 경우 게스트 계정으로 로그인 하는경우 다른 게스트 계정을 생성합니다.

- 설정 옵션
  - `configuration` : AppUserIdentityGenerator.class
  - json 설정 파일 : app

Instant 식별자 생성

휘발성 게스트 계정을 생성합니다. 이 모드로 앱을 설정하면 게스트를 로그아웃 할때 게스트 계정을 잃어버리게 됩니다.

- 설정 옵션
  - `configuration` : InstantUserIdentityGenerator.class
  - json 설정 파일 : instant

세션 매니져 생성

EgClient의 `getSessionManager()` 메소드를 사용하여 `SessionManager` 을 생성합니다. `SessionManager` EG 플랫폼의 세션 및 토큰의 제어 및 조회 기능을 담당합니다.

- 세션 생성. (게스트 세션 생성)
- 세션, 유저정보 조회
- 로그아웃

```
final SessionManager sessionManager = client.getSessionManager();
```

세션 확인 메소드

함수명	리턴값	설명
isSessionOpen	boolean	세션의 만료 여부와는 상관없이 생성된 세션 (발급된 토큰)이 있으면 True를 반환합니다.
getToken	Token	토큰정보를 반환합니다.
getProfile	Profile	사용자 프로필 정보를 반환합니다.

세션 제어 메소드

함수명	설명	리턴값	매개변수
open	게스트 생성, 계정이 있을 경우는 토큰 갱신	com.estgames.framework.core.Task	없음
create	게스트 계정 생성	com.estgames.framework.core.Task	없음
resume	토큰 갱신	com.estgames.framework.core.Task	없음
expire	세션 토큰 만료	com.estgames.framework.cor.Task	없음
signOut	로그아웃	com.estgames.framework.core.Task	없음

세션 생성(게스트 로그인) 및 세션 갱신

세션 생성은 create() 메소드로 만드며 해당 메소드는 Task 클래스를 리턴합니다. Task는 빌더 패턴을 이용하여 로그인이 끝난 이후 혹은 에러 시 처리를 구현합니다.

앱을 시작했을 때 세션이 없을 경우에는 create() 메소드를 사용하여 게스트 계정을 만들 수 있고 현재 세션이 있는 상태라면(SNS계정 포함) 세션을 다시 사용하기 위해 갱신을 해주어야 합니다. 갱신은 resume() 메소드를 호출하여 토큰을 갱신할 수 있습니다.

Task에서 제공하는 메소드에서는 하나의 함수를 구현하는 인터페이스인 Acceptor를 매개변수로 받습니다.

Task에서 제공하는 메소드

함수명	설명	리턴값	매개변수
asyncAccept	정상종료 이후 처리	com.estgames.framework.core.Task	Acceptor<T>
onError	에러시 처리 구현	com.estgames.framework.session.Token	Acceptor<T>

예) 게스트 세션생성 및 생성 이후, 실패 시 처리

```
sessionManager
    .create()
    .onError(new Task.Acceptor<Throwable>() {
        @Override
        public void accept(Throwable t) {
            // 세션 생성 실패 핸들러 작성
        }
    })
    .asyncAccept(new Task.Acceptor<String>() {
        @Override
        public void accept(String token) {
            // 세션 생성 완료 핸들러 작성
        }
    });
```

예) 세션 갱신

```
sessionManager
    .resume()
    .onError(new Task.Acceptor<Throwable>() {
        @Override
        public void accept(Throwable throwable) {
            currentTitle.setText("세션 resume 실패 = " + throwable.getMessage());
        }
    })
    .asyncAccept(new Task.Acceptor<String>() {
        @Override
        public void accept(String s) {
            currentTitle.setText("세션 resume 성공 = " + s);
        }
    });
```

세션, 유저정보 조회하기

SessionManager에서 getToken() 메소드로 현재 세션의 토큰 정보를 조회할 수 있으며 getProfile() 메소드로 프로필 정보를 조회할 수 있습니다.

SessionManager 세션정보 조회관련 메소드

함수명	리턴타입	설명
getToken	com.estgames.framework.session.Token	토큰정보
getProfile	com.estgames.framework.session.Profile	유저관련 정보

Token

함수명	리턴타입	설명
getEgToken	String	EgToken
getRefreshToken	String	RefreshToken

Profile

함수명	리턴값	설명
getEgId	String	EgId
getUserId	String	유저아이디
getProvider	String?	프로바이더(gmail,facebook)
getEmail	String?	이메일
getPrincipal	String	Principal

예)

```
Token token = client.getSessionManager().getToken(); // 현재 세션의 토큰정보 조회
Profile profile = client.getSessionManager().getProfile(); // 현재 세션의 프로필정보 조회하기

if (token != null) {
    egTokenTitle.setText("getEgToken = " + token.getEgToken());
    refreshTokenTitle.setText("getRefreshToken = " + token.getRefreshToken());
}

if (profile != null) {
    egIdTitle.setText("getEgId = " + profile.getEgId());
    emailTitle.setText("getEmail = " + profile.getEmail());
    pricipalTitle.setText("getPrincipal = " + profile.getPrincipal());
    providerTitle.setText("getProvider = " + profile.getProvider());
    userIdTitle.setText("getUserId = " + profile.getUserId());
}
```

로그아웃하기

`SessionManager` 에 있는 signOut 메소드로 로그아웃을 진행합니다.

```
sessionManager.signOut().asyncAccept(new Task.Acceptor() {
    @Override public void accept(Object o) {
        // 세션 종료 핸들러 작성
        System.out.println("로그아웃");
    }
});
```

SNS 계정연동

플랫폼 SDK는 SNS 계정에 로그인 하고 플랫폼 계정과 연동하는 역할을 하는 `SignInControl` 클래스를 제공합니다.

현재 플랫폼이 제공하는 SNS 계정연동은 Facebook 과 Google 입니다. SNS 계정연동은 Android Framework 에서 제공하는 순수 컴포넌트만을 이용할 수도 있고, EG 플랫폼에서 제공하는 로그인 버튼 컴포넌트나 로그인 대화창 (Activity 기반)을 이용할 수도 있습니다.

계정연동은 다음과 같은 순서로 할 수 있습니다.

기본 버튼으로 로그인

1. 레이아웃에 버튼 배치

```
<Button
    android:id="@+id/btn_facebook_login"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:text="Facebook Login"/>
```

2. SignInControl 설정 (in Activity)

`SignInControl` 객체는 `onActivityResult` 콜백에 등록해야 합니다. 따라서 인스턴스 변수로 생성해야합니다.

```
// Sign in option 설정
SignInControl.Option option = new SignInControl.Option()
        .setSignInResultHandler(new SignInResultHandler() {
            @Override
            public void onComplete(Result.Login result) {
                // 로그인 성공 핸들러 작성
            }

            @Override
            public void onError(Throwable t) {
                // 로그인 실패 핸들러 작성
            }

            @Override
            public void onCancel() {
                // 로그인 취소 핸들러 작성
            }
        });

// SignInControl 객체생성
SignInControl signInControl = SignInControl.createControl(activity, option);
```

3. Button 핸들러에 연결

로그인 버튼을 눌렀을때 로그인을 시도하도록 핸들러에 등록합니다.

```
Button facebookLogin = findViewById(R.id.btn_facebook_login);
facebookLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        signInControl.signIn(Provider.FACEBOOK, activity);
```

```
    }
});
```

4. ActivityResult 핸들러 작성

로그인 결과를 핸들러에 전달 할 수 있도록 Activity.onActivityResult 메소드를 작성합니다.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    signInControl.onActivityResult(requestCode, resultCode, data);
}
```

EG SDK 제공 버튼으로 로그인

1. 레이아웃에 버튼 배치

```
<com.estgames.framework.ui.buttons.FacebookSignButton
        android:id="+@id/btn_facebook_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

<com.estgames.framework.ui.buttons.GooogleSignButton
        android:id="+@id/btn_google_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
```

2. SignInControl 설정 (in Activity)

`SignInControl` 객체는 `onActivityResult` 콜백에 등록해야 합니다. 따라서 인스턴스 변수로 생성해야합니다.

레이아웃 xml 파일에 명시한 `SignButton` 을 Control 객체와 연결합니다.

```
// Sign in option 설정
SignInControl.Option option = new SignInControl.Option()
        .addSignButton((SignButton) findViewById(R.layout.btn_facebook_login))
        .addSignButton((SignButton) findViewById(R.layout.btn_google_login))
        .setSignInResultHandler(new SignInResultHandler() {
            @Override
            public void onComplete(Result.Login result) {
                // 로그인 성공 핸들러 작성
            }

            @Override
            public void onError(Throwable t) {
                // 로그인 실패 핸들러 작성
            }

            @Override
            public void onCancel() {
                // 로그인 취소 핸들러 작성
            }
        });

// SignInControl 객체생성
SignInControl signInControl = SignInControl.createControl(activity, option);
```

3. ActivityResult 핸들러 작성

로그인 결과를 핸들러에 전달 할 수 있도록 Activity.onActivityResult 메소드를 작성합니다.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    signInControl.onActivityResult(requestCode, resultCode, data);
}
```

EG SDK 제공 대화창(SignInActivity)으로 로그인

1. SignInControl 설정 (in Activity)

`SignInControl` 객체는 `onActivityResult` 콜백에 등록해야 합니다. 따라서 인스턴스 변수로 생성해야합니다.

레이아웃 xml 파일에 명시한 `SignButton` 을 Control 객체와 연결합니다.

```
// Sign in option 설정
SignInControl.Option option = new SignInControl.Option()
        .setSignInResultHandler(new SignInResultHandler() {
            @Override
            public void onComplete(Result.Login result) {
                // 로그인 성공 핸들러 작성
            }

            @Override
            public void onError(Throwable t) {
                // 로그인 실패 핸들러 작성
            }

            @Override
            public void onCancel() {
                // 로그인 취소 핸들러 작성
            }
        });

// SignInControl 객체생성
SignInControl signInControl = SignInControl.createControl(activity, option);
```

2. ActivityResult 핸들러 작성

로그인 결과를 핸들러에 전달 할 수 있도록 Activity.onActivityResult 메소드를 작성합니다.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    signInControl.onActivityResult(requestCode, resultCode, data);
}
```

3. 로그인 대화창 호출

```
signInControl.signIn(activity);
```

로그인 컨트롤

`SignInControl` 클래스는 외부 프로바이더 계정의 로그인 과 EG 플랫폼 계정의 연동을 담당합니다. `SignInControl` 클래스를 통해 SNS 계정 연동을 간편하게 진행 할 수 있습니다.

객체 생성

`SignInControl` 클래스는 옵션을 입력받는 factory 메소드를 가지고 있습니다. 이 factory 메소드를 이용해 객체를 생성합니다.

Option

`SignInControl` 객체를 생성을 위한 옵션을 설정 합니다.

옵션설정 값은 다음과 같습니다.

옵션이름	메소드	파라미터 타입	값	설명
MODE	setMode()	String	SYNC(default), SWITCH	외부 프로바이더 인증 후 연동 방법을 설정합니다. 값은 `sync` , `switch` 입니다. `switch` 모드는 계정을 변경하여 세션을 생성합니다. 대소문자를 가리지 않습니다.
SignButton	addSignButton()	SignButton	-	로그인 버튼을 Control 객체와 연결합니다. Control 객체와 연결된 버튼은 따로 `OnClickHandler` 를 등록하지 않아도 됩니다.
SyncAware	setAccountSyncAware()	AccountSyncAware	-	계정 연동시 계정 충돌 이벤트 핸들러를 등록합니다. 계정 충돌이 발생한 경우 처리를 위해 사용합니다. 따로 등록하지 않는 경우 SDK 기본핸들러가 등록됩니다.
ResultHandler	setSignInResultHandler()	SignInResultHandler	-	로그인 결과 핸들러를 등록합니다.

```
SignInControl.Option option = new SignInControl.option()
        .setMode("sync")          // 프로바이더 계정 연계 모드. 기본값은 SYNC (연동) 모드입니다.
        .addSignButton( ... )     // 로그인 버튼 등록.
        .setAccountSyncAware(new AccountSyncAware() {
            // 로그인 후 계정연결시 충돌상황에 대한 핸들러 등록
        })
        .setSignInResultHandler(new SignInResultHandler() {
            // 로그인 결과 핸들러 등록
        });
```

Factory 메소드

`SignInControl` 객체를 생성합니다.

```
SignInControl control = SignInControl.createControl(context, option);
```

메소드

`SignInControl` 객체 메소드들입니다.

이름	파라미터	리턴값	설명
isSignIn	Provider	Boolean	파라미터로 받은 프로바이더 계정의 로그인 여부를 확인합니다.
signIn	Provider, Activity	-	파라미터로 받은 프로바이더 계정으로 로그인을 합니다. `SignInControl` 객체에 버튼을 연결하지 않고 버튼을 직접 구현하는 경우 사용합니다.
signIn	Activity	-	SDK에서 기본적으로 제공하는 로그인 팝업창을 호출합니다.
onActivityResult	RequestCode(int), ResultCode(int), Data(Intent)	-	로그인 결과 처리를 위한 콜백들을 호출합니다. 로그인 처리를 위해 반드시 `Activity` 의 `onActivityResult` 메소드에서 호출 해줘야 합니다.

핸들러

AccountSyncAware

계정 연동중 계정 충돌 이벤트를 제어하기 위한 핸들러 인터페이스입니다. 계정 충돌이 발생한경우 연동을 계송 진행할지 중단 할지 여부를 선택 할 수 있습니다.

이름	파라미터	설명
preProcess	-	계정 연동을 하기 직전에 호출됩니다. 연동중 화면 보호를 위한 스크린이나 ProgressBar를 보여주는 용도등으로 사용할 수 있습니다. Hook 메소드이므로 반드시 구현할 필요는 없습니다.
postProgress	-	계정 연동이 완료되거나 실패한 직후에 호출 됩니다. 연동 직전에 보여준 ProgressBar를 회수하는 용도등으로 사용할 수 있습니다. Hook 메소드이므로 반드시 구현할 필요는 없습니다.
onMismatched	Mismatch, Step	계정 연동중 계정 충돌이 발생한 경우에 호출됩니다. 충돌한 계정 정보와 다음 프로세스의 진행 여부를 선택 할 수 있습니다.

계정 연동중 계정 충돌이 발생한 경우 `onMismatched` 메소드가 호출됩니다. 이때 SDK 는 `Mismatch` 객체와 `Step` 객체를 넘겨주게 됩니다.

Mismatch

충돌이 발생한 계정 정보를 포함하고 있습니다.

속성 이름	데이터 타입	설명
conflictId	String	충돌이 발생한 EG ID 값입니다. 현재 세션의 EG ID 값이 아닙니다.
user	String	Provider 계정의 사용자 정보입니다. 구글과 페이스북의 경우 사용자 Email 계정입니다.
data	Map	Provider 계정의 상세 정보입니다. key - value 형태의 값으로 넘겨줍니다.

Step

다음 프로세스의 처리 여부를 SDK에 전달합니다.

메소드 이름	파라미터	설명
stop	-	계정 연동을 중단합니다. 이 경우 ResultHandler 에 취소로 값을 전달합니다.
stop	Throwable	계정 연동을 중단합니다. 이경우 ResultHandler 에 에러로 값을 전달합니다.
proceed	-	충돌이 발생했으나 강제로 계정 연동을 진행합니다. 충돌한 계정의 정보는 삭제됩니다. 이 메소드는 비동기로 동작합니다.
proceedAndAwait	-	충돌이 발생했으나 강제로 계정연동을 진행합니다. 충돌한 계정의 정보는 삭제됩니다. 이 메소드는 동기모드로 동작합니다. 이 후에 반드시 `done()` 메소드를 호출 해야 합니다.
done	-	모든 프로세스를 완료합니다. 이 메소드는 ResultHandler 에 값을 전달합니다. 이 메소드를 호출 하면 동기모드로 진행 할 경우에만 호출 해 주면 됩니다.

`proceed()` 메소드의 경우 동기와 비동기모드로 선택하여 진행 할 수 있습니다. 비동기 모드로 진행 할 경우 모든 프로세스가 완료되면 자동으로 ResultHandler에 값을 전달합니다. 그러나 동기 모드로 진행할 경우 `done()` 메소드를 호출하여 프로세스 완료 처리를 해주어야 합니다. `stop()` 의 경우에는 중단과 동시에 완료처리를 합니다.

SignInResultHandler

로그인 결과를 전달 받는 핸들러 인터페이스입니다. Provider 계정 연동 후 결과를 처리하기 위해 사용합니다.

메소드 이름	파라미터	설명
onComplete	Result.Login	계정연동이 성공한 경우 호출됩니다.
onError	Throwable	계정연동이 실패한 경우 호출됩니다. 실패한 예외 객체를 넘겨받습니다.
onCancel	-	계정연동을 취소한 경우 호출됩니다.

Result.Login

성공한 계정 연동 정보를 포함하는 데이터 객체입니다.

속성이름	데이터 타입	설명
type	String	계정연동 유형입니다. LOGIN, SYNC 값으로 어떤 방식으로 로그인 했는지 판단합니다.
egId	String	계정을 연동한 후 EG ID. 현재 세션의 EG ID 와 동일합니다.
provider	String	계정 연동을 진행한 프로바이더 값입니다.

상품 구매 및 결제

EG 모바일 플랫폼의 안드로이드 SDK는 구글 플레이 인앱 상품 결제를 지원합니다.

- 상품 구매 및 결제하기

결제 컨트롤

`PaymentControl` 은 아이템 구매 및 결제 서비스를 담당합니다. EG 플랫폼의 안드로이드 SDK는 구글 플레이의 인앱 서비스를 지원하고 있습니다.

객체 생성

`PaymentControl` 클래스는 옵션을 입력받는 factory 메소드를 가지고 있습니다. 이 factory 메소드를 이용해 객체를 생성합니다.

```
// 구글 플레이 인앱 결제를 위한 Payment Control 객체를 생성합니다.
// 객체를 생성하기 위해서는 옵션을 설정해야 합니다.
PaymentControl.Option option = new PaymentControl.Option()
        .implicitRestore(true)
        .platformApiKey(context.getString(R.string.google_payment_public_key))
        .restoreListener(new ProcessListener<String[]>() {
            @Override public void onComplete(String[] restores) {
                // restore 목록 조회 완료 핸들러 작성
                // 미지급된 아이템(상품)의 order id 를 Array 로 받습니다.
            }

            @Override public void onError(@NotNull Throwable t) {
                // restore 목록 조회 실패 핸들러 작성
            }
        })
        .purchaseListener(new ProcessListener<String>() {
            @Override
            public void onComplete(String orderId) {
                // 구매 완료 핸들러 작성
            }

            @Override public void onError(@NotNull Throwable t) {
                // 구매 실패 핸들러 작성
            }
        });

PaymentControl payControl = PaymentControl.createControl(context, option);
```

PaymentControl.Option

구매 Control 객체를 생성하기 위한 옵션은 다음과 같습니다.

implicitRestore

- implicitRestore 옵션은 Control 객체 생성시 자동으로 구매 복구 프로세스 실행 여부를 결정합니다.
- 이 옵션의 기본값은 `true` 입니다. 이 옵션이 활성화 되어 있으면 `PaymentControl` 객체가 생성되고 난 후 자동으로 미지급 아이템(상품) 목록을 조회하여 클라이언트에 전달합니다.

platformApiKey

- 결제 플랫폼의 API 인앱 결제 공개키 값입니다.
- Google Play Console 의 경우 API 인앱 결제 공개키를 생성 및 발급해 줍니다. 클라이언트는 이 공개키를 이용해 사용자를 인증하게 됩니다. 따라서 이 값은 반드시 설정해야 합니다.
- 인앱 결제 공개키는 앱 내에서 안전하게 관리되어야 합니다.

restoreListener

- Control 객체에서 조회한 restore 아이템(상품) 목록에 대한 처리를 위한 핸들러를 등록합니다.
- 콜백으로 넘어오는 인자는 미지급된 아이템(상품)의 `orderId` 를 Array 로 받습니다.
- 이 핸들러에서 미지급된 아이템에 대한 지급 처리를 합니다.

purchaseListener

- Control 객체로 구매한 아이템(상품)에 대한 처리를 위한 핸들러를 등록합니다.
- 콜백으로 넘어오는 인자는 구매한 아이템(상품)에 대한 `orderId` 입니다.
- 이 핸들러에서 아이템에 대한 지급 처리를 합니다.

Product Restore

미지급된 구매 아이템(상품) 복구

앱에서 결제한 아이템은 게임 서버를 통해 지급 처리를 요청하게 됩니다. 이때 불안정한 네트워크등 여러 상황에 따라 구매한 아이템이 미지급 될 수도 있습니다. 어떠한 상황이든 사용자가 구매한 아이템에 대해 확실한 지급 처리를 보장해야 합니다.

EG 플랫폼 SDK의 구매 모듈은 Control 객체가 생성되고 로드될때 암묵적으로 미지급 아이템들을 처리하는 기능을 제공하고 있습니다.

암묵적 Restore 기능은 `PaymentControl` 객체를 생성할때 `implicitRestore` 옵션으로 설정 할 수 있으며 기본값은 `true` 입니다. 이 옵션값을 `false` 로 지정할 경우 `restore()` 메소드를 호출 함으로써 미지급 아이템에 대한 지급 처리를 진행 할 수 있습니다.

```
pymentControl.restore();        // 미지급된 아이템 목록을 처리합니다.
```

Purchase

아이템(상품) 구매

아이템의 구매는 `PaymentControl` 객체의 `purchase()` 메소드를 통해 진행 할 수 있습니다.

purchase

- 입력 파라미터 : productId - 구글 플레이 인앱에 등록된 상품 코드입니다.

```
paymentControl.purchase(activity, "product.id");
```

onActivityResult

아이템을 구매하기 위해서는 결제 정보 입력, 구매 동의등의 사용자와 상호작용이 필요합니다. 따라서 구매 완료를 처리하기 위해 result callback 을 등록해야 합니다. Control 객체의 콜백은 `onActivityResult` 메소드에 등록해 주면 됩니다.

```
public void onActivityResult(int requestCode, int responseCode, Intent data) {
    ...
    paymentControl.onActivityResult(requestCode, responseCode, data);
}
```

Product 조회

아이템(상품) 조회

앱 클라이언트는 `PaymentControl` 객체의 `product()` 메소드를 통해 인앱에 등록된 상품 정보를 조회 할 수 있습니다.

product

- 입력 파라미터 : productId - 구글 플레이 인앱에 등록된 상품 코드입니다.
- Return : TaskData - 상품 조회는 기본적으로 비동기적으로 작동합니다.

```
TaskData<Product> product = paymentControl.product("product.id")
product.getAsync(new AsyncReceiver<Product> {
    @Override public void receive(Product product) {
        // ... 상품정보를 처리하는 코드
    }
});
```

> 상품 정보 조회를 동기적으로 처리하고 싶은 경우 `TaskData` 의 `get()` 메소드를 사용하세요. `Future` 와 동일하게 동작합니다.

```
Product product = paymentControl.product("product.id").get();
```