|  |  |  |  |
| --- | --- | --- | --- |
| **FINAL PROJECT 개발 보고서** | | | |
| **COUSRE** | **KITRI 이공계 전문기술연수 과정 Intern 50th** | | |
| **SUBJECT** | **KITRI FINAL PROJECT** | | |
| **TITLE** | **술친구 주선자(HelloDrinking)** | | |
| **P.M.** | 전 재 형 | **Team Members** | 전재형, 정길수, 정찬규 |
| **Principal Engr.** | 김 진 석 | **Duration** | 2011.07.18. ~ 2011.09.02. |
| 한국정보기술연구원(KITRI)의 프로젝트 수행 규정에 의거하여  2011년 이공계기술연수과정 50차 **서비스지향 오픈프레임워크 기반 애플리케이션 개발 과정** 3 조의 최종 프로젝트 개발 보고서를 아래와 같이 제출합니다.  **P.M. - 키 트 리 (인)**  2011년 09월 02일  **한 국 정 보 기 술 연 구 원 장 귀 하** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **FINAL PROJECT 개발예정 보고서** | | | |
| **COUSRE** | **이공계 전문기술연수과정 50th** | | |
| **SUBJECT** | **KITRI FINAL PROJECT** | | |
| **TITLE** | **술친구 주선자** | | |
| **P.M.** | 정 찬규 | **Team Members** | 정 길수, 정 찬규, 전 재형 |
| **Principal Engr.** | 김 진석 | **Duration** | 2011.07.18 ~ 2011.09.02. |
| 한국정보기술연구원(KITRI)의 프로젝트 수행 규정에 의거하여  2011년 이공계 전문기술연수과정 서비스지향 오픈프레임워크 기반 프로젝트 개발과정 3 조의 최종 프로젝트 개발예정 보고서를 아래와 같이 제출합니다.  **P.M. - 정 찬 규 (인)**  2011년 07월 11일  **한 국 정 보 기 술 연 구 원 장 귀 하** | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1. 프로젝트 개요**  **위치 정보를 바탕으로 만남을 제안하는 안드로이드 어플리케이션 제작.**  **- 개발 동기 및 필요성**  **1. 안드로이드 폰 사용자 증가**  **안드로이드 사용자수**  **2. 즉석만남의 이슈 및 관심 증가**  네이버    뉴스에서의  '  즉석만남  '  연도별 검색 건수  2  8  11  12  17  26  32  54  154  172  244  327  0  50  100  150  200  250  300  350  2000  2001  2002  2003  2004  2005  2006  2007  2008  2009  2010  2011  **3. 즉석만남을 하기 위한 리스크**   * **마음에 드는 상대를 찾는데 소요되는 비용이 크다.** * **상대의 정보가 없다.**   **- 목표 : 즉석 만남의 기회 및 편의를 제공하기 위함**  **1. 장소 정보를 제공**  **2. 위치 계측 및 방향성에 대한 쉬운 접근을 제공**  **3. 위치 기반 서비스의 활용에 대해 학습**  **4. 완성도 있는 프로그램의 제작 및 관련 보고서 작성**  **5. 프로그램 판매를 통한 수익 창출 및 수익 구조에 대한 체험을 목표**  **- 경쟁 상품**   1. **1Km : 아이폰 어플리케이션**  |  |  | | --- | --- | | **mzl** | * **현재 자신의 위치에서 특정 거리 내의 사용자를 검색** * **자신의 위치를 기초로 불특정 상대방을 검색** * **사진 위주의 정보를 제공** * **프로필 정보를 기초로 상대방을 검색 가능** |  1. **Hi There : 안드로이드 어플리케이션**  |  |  | | --- | --- | | **새 비트맵 이미지** | * **가입된 사용자 모두에게 메시지를 전송** * **자신의 위치를 기초로 불특정 상대방을 검색** * **지도에서 타 사용자의 위치 정보를 확인 가능** * **개인 블로그 생성 및 이용** |   **2. 개발 환경**  **Relay Server**  **DataBase**  **Mobile Node**  Oracle XE  TCP/IP (1521)  TCP/IP (18080)  JRE 1.6  <<exeuctionEvironment>>  **ClientAcceptThread**  **DAO**  use  **Naver Map Api** 개발 도구  * + **eclipse-java-helios-SR2-win32 를 기본 개발 언어로 사용한다.**   + **Java JDK 1.6 을 Java 프로그램 개발 Kit로 사용한다.**   + **Android SDK 2.1 를 Android 개발 Kit로 사용한다.**   + **Map 정보 활용을 위해 Naver Map Api를 사용한다.**     - http://dev.naver.com/openapi/apis/map/android/example   + **Motorola Motoroi (Android version: 2.1)을 기본 테스트기로 사용한다.**   + **Oracle 10g XE 를 Database로, SQL developer를 Database 관리 도구로 사용한다.**   + **Git(eGit 1.0.0) 을 형상 관리 툴로서 사용한다.**     - http://www.eclipse.org/egit/   + **그 외 설계를 위해 StarUML, ERWin 등 을 사용한다.** * **역할 분담**  |  |  | | --- | --- | | 이름 | 역할 | | 전재형 | 프로젝트 매니저, Server Client DB 기능 설계 및 구현 | | 정길수 | Client UI 설계 및 구현 | | 정찬규 | Server DB 기능 설계 및 구현 |  * **개발 일정**  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Time  Line  Mission | **2011. 07. 18** | **2011. 07. 19** | **2011. 07. 20** | **2011. 07. 21** | **2011. 07. 22** | **2011. 07. 25** | **2011. 07. 26** | **2011. 07. 27** | **2011. 07. 28** | **2011. 07. 29** | | **기능 분석** |  |  |  |  |  |  |  |  |  |  | | **시스템 명세** |  |  |  |  |  |  |  |  |  |  | | **기능 명세** |  |  |  |  |  |  |  |  |  |  | | **DB 명세** |  |  |  |  |  |  |  |  |  |  | | **DB 구축** |  |  |  |  |  |  |  |  |  |  | | **Server 설치** |  |  |  |  |  |  |  |  |  |  | | **Server 분석논리)** |  |  |  |  |  |  |  |  |  |  | | **Server 설계(물리)** |  |  |  |  |  |  |  |  |  |  | | **Server 구현** |  |  |  |  |  |  | | |  |  | | **Client 분석(논리)** |  |  |  |  |  |  |  |  |  |  | | **Client 설계(물리)** |  |  |  |  |  |  |  |  |  |  | | **Client 구현** |  |  |  |  |  |  | | |  |  | | **Server-Client  통신 연계** |  |  |  |  |  |  |  |  |  |  | | **Debugging** |  |  |  |  |  |  |  |  |  |  | | **결과 보고서 및  발표 자료 제작** |  |  |  |  |  |  |  |  |  |  |   **3. 시스템 설계 (실제 개발된 시스템에 대한 설계 내용을 기술한다)**  **[시스템 개요]**   1. **필수 기능**  * **회원 가입 및 로그인** * **게시 정보를 입력** * **메시지 전달 및 수신** * **위치 정보를 통한 지도 표현** * **위치 정보를 바탕으로 증강 현실 구현**  1. **사용 개요**    1. **만남을 원하는 사용자가 관련 정보를 게시한다.**    2. **자신의 위치 정보를 바탕으로 하여, 거리가 가까운 순서로 만남을 원하는 사용자가 게시한 정보를 리스트 형태로 게시한다.**        * **Map, 증강현실 형태의 정보 보기로 전환가능 하다.**    3. **게시된 정보를 통해 마음에 드는 상대에게 메시지를 보낸다.**    4. **메시지를 받은 이용자는 상대방의 정보를 확인한 후, 허락 여부를 결정하여 연락을 취한다.** 2. **뷰 개요**  |  |  | | --- | --- | | **리스트** | * **게시정보: 자신 및 일행을 홍보** * **게시정보를 통해 게시자의 프로필을 확인 가능** | | **MAP**  **검색조건 리스트 증강현실** | * 지도에서 타 사용자의 위치 확인 * 상점 등 주변 정보 확인 | | **증강현실** | * 카메라 모드를 기초로 타 사용자의 방향과 거리를 표현 |  1. **모델 개요**  * **개인 정보**   + **필수 기재 사항:** 아이디, 암호, 성별   + **선택 기재 사항:** 이름, 나이, 사진, 홈페이지, 주소, 전화번호 * **게시 정보**   + **필수 기재 사항 :** 사진, 인원, 위치   + **선택 기재 사항 :** 나이, 주량, 코멘트 |

목차

[- 개발 도구 5](#_Toc302622343)

[1. 서론 14](#_Toc302622344)

[1.1. 개발의 필요성 및 목적 14](#_Toc302622345)

[1.1.1. 개발의 필요성 14](#_Toc302622346)

[1.1.2. 개발의 목적 15](#_Toc302622347)

[1.1 개발의 범위 및 방법 15](#_Toc302622348)

[1.1.3. 역할 분담 15](#_Toc302622349)

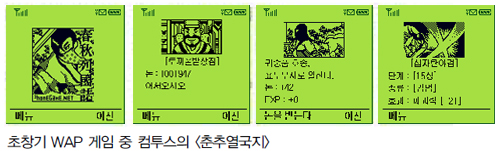
[1.1.4. 개발 일정: 수정 필요 15](#_Toc302622350)

[2. 주제에 대한 이론적 배경 15](#_Toc302622351)

[2.1. 배경 15](#_Toc302622352)

[2.1.1. 과거의 PC는 랩탑 컴퓨터라 하더라도 책상에 앉아서 주로 사용하였다. 15](#_Toc302622353)

[2.1.2. 기술이 발전됨에 따라 소프트웨어가 할 수 있는 일의 범위를 더욱 넓혀주었다. 15](#_Toc302622354)

[2.1.3.  15](#_Toc302622355)

[2.1.4.  16](#_Toc302622356)

[2.1.5.  16](#_Toc302622357)

[2.1.6. 스마트폰으로 인해 가지고 다니는 컴퓨터가 확산 되었고 LBS와 GPS로 인해 자신의 위치정보를 실시간으로 서버에 전달할 수 있게 되었다. 16](#_Toc302622358)

[2.2. 특징 또는 장단점, 특성 유형과 형태 17](#_Toc302622359)

[2.2.1. 장단점 17](#_Toc302622360)

[2.2.2. 특성 유형과 형태 18](#_Toc302622361)

[사례분석 19](#_Toc302622362)

[**사례에 대한 현황** 19](#_Toc302622363)

[**사례분석 및 실증조사** 22](#_Toc302622364)

[2.2.3. 현황 및 조사대상, 조사방법 등 22](#_Toc302622365)

[2.2.4. 사례에 대한 정리 23](#_Toc302622366)

[프로젝트 수행내용 27](#_Toc302622367)

[2.3. 수행범위 27](#_Toc302622368)

[2.3.1. 개발 환경 27](#_Toc302622369)

[2.3.2. 개요 28](#_Toc302622370)

[2.3.3. DB 모델링 30](#_Toc302622371)

[2.3.4. 설치법 31](#_Toc302622372)

[2.3.5. 시스템 아키텍처 모델 35](#_Toc302622373)

[2.4. Achitecture Model 35](#_Toc302622374)

[2.5. 개발 환경 37](#_Toc302622375)

[2.5.1. 개발툴 37](#_Toc302622376)

[2.5.2. 버전관리계획 37](#_Toc302622377)

[2.5.3. 데이터베이스 접근 계획 37](#_Toc302622378)

[2.6. 명명 규칙 39](#_Toc302622379)

[2.6.1. 디렉토리 명명 규칙 39](#_Toc302622380)

[2.6.2. Class 및 interface 명명 규칙 39](#_Toc302622381)

[2.6.3. Class 내부의 변수 및 메서드 명명 규칙 39](#_Toc302622382)

[2.6.4. 리소스 id의 명명 규칙 40](#_Toc302622383)

[2.6.5. 데이터베이스 명명 규칙 40](#_Toc302622384)

[2.7. 개발 구조 42](#_Toc302622385)

[2.7.1. 디렉토리 구조 42](#_Toc302622386)

[2.7.2. 공용 라이브러리 구조 42](#_Toc302622387)

[2.8. 기능 요구서 43](#_Toc302622388)

[2.9. 데이터베이스 45](#_Toc302622389)

[2.9.1. USERINFO : 가입된 사용자의 정보 테이블 45](#_Toc302622390)

[2.9.2. POSTINFO : 게시물의 정보 테이블 47](#_Toc302622391)

[2.9.3. LOGINHISTORY : 로그인 시도 기록 테이블 48](#_Toc302622392)

[2.9.4. 사용 Sequence 49](#_Toc302622393)

[2.10. Mobile Service 50](#_Toc302622394)

[2.10.1. 디렉토리 구성 50](#_Toc302622395)

[2.10.2. 패키지 구성 50](#_Toc302622396)

[2.11. Service Server 71](#_Toc302622397)

[2.11.1. 디렉토리 구성 71](#_Toc302622398)

[2.11.2. 패키지 구성 71](#_Toc302622399)

[1.2 결론 87](#_Toc302622400)

[2.12. 미 해결 문제점 87](#_Toc302622401)

[1.3 참고자료 88](#_Toc302622402)

[2.13. 네이버 지도 API : http://dev.naver.com/openapi/apis/map/android/example 88](#_Toc302622403)

[2.14. 센서 카메라 위치정보 활용 - 박유태 88](#_Toc302622404)

[2.15. Mixare : http://code.google.com/p/mixare/ 88](#_Toc302622405)

[2.16. Wikitude API 1.1 : http://www.wikitude.org 88](#_Toc302622406)

[2.17. hi there : 안드로이드 마켓 88](#_Toc302622407)

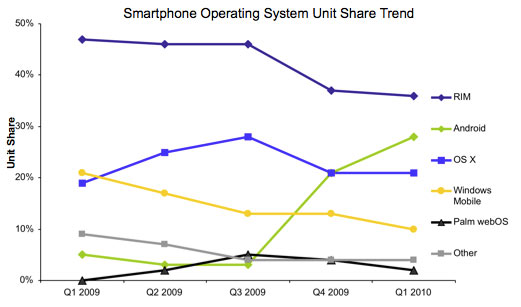
[2.18. 이미지 참고 자료 88](#_Toc302622408)

# 서론

# 개발의 필요성 및 목적

### 개발의 필요성

* 안드로이드 폰 사용자 증가



* 즉석만남의 이슈 및 관심 증가

네이버 뉴스에서 `즉석만남’연도별 검색 건수

2

MapContainerView

11

CompassView

17

CameraView

32

54

UserInfoDialog

ConfigureDialog

244

327

0

100

150

200

250

300

350

00

01

02

03

04

05

06

07

08

09

10

11

* 즉석만남을 위한 리스크
  1. 마음에 드는 상대를 찾는데 소요되는 비용이 크다.
  2. 상대의 정보가 없다.

### 개발의 목적

- 목표 : 즉석 만남의 기회 및 편의를 제공하기 위함

 장소 정보를 제공

 위치 계측 및 방향성에 대한 쉬운 접근을 제공

 위치 기반 서비스의 활용에 대해 학습

 완성도 있는 프로그램의 제작 및 관련 보고서 작성

 프로그램 판매를 통한 수익 창출 및 수익 구조에 대한 체험을 목표

# 개발의 범위 및 방법

### 역할 분담

|  |  |
| --- | --- |
| 이름 | 역할 |
| 전재형 | 프로젝트 매니저, Server Client DB 기능 설계 및 구현 |
| 정길수 | Client UI 설계 및 구현 |
| 정찬규 | Server DB 기능 설계 및 구현 |

### 개발 일정: 수정 필요

# 주제에 대한 이론적 배경

# 배경

### 과거의 PC는 랩탑 컴퓨터라 하더라도 책상에 앉아서 주로 사용하였다.

### 기술이 발전됨에 따라 소프트웨어가 할 수 있는 일의 범위를 더욱 넓혀주었다.

### 소프트웨어%20변천사1

< 그림 4 > 그래픽 성능의 부재로 인해 모바일 게임은 머드게임에 가까웠었다.

### 소프트웨어%20변천사2

< 그림 5 > 칼라휴대폰이 나오고 좀더 컨텐츠가 다양해졌다.

### 소프트웨어 변천사3

< 그림 6 > 네트워크를 통해 상대방과 대전이 가능하게 되었다.

### 스마트폰으로 인해 가지고 다니는 컴퓨터가 확산 되었고 LBS와 GPS로 인해 자신의 위치정보를 실시간으로 서버에 전달할 수 있게 되었다.

****

< 그림 7 > 법적인 문제로 정지 당했던 ‘오빠믿지’ App 이다.

# 특징 또는 장단점, 특성 유형과 형태

### 장단점

#### 장점

#### 모든 연령대가 사용하는 만남주선을 목적으로 하는 타 어플리케이션과 달리 성인만을 목적으로 하고 있기에 좀 더 개방적인 게시글을 올릴 수 있다.

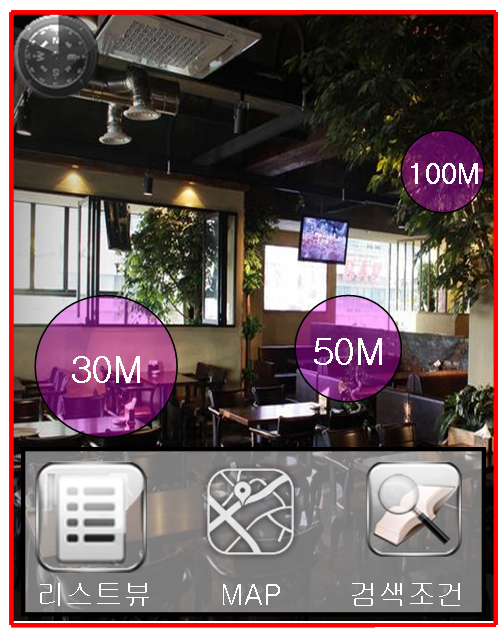
#### 리스트뷰

< 그림 8 > 타 유사 프로그램과 비슷한 인터페이스인 술친구 주선자의 리스트 화면

#### 단점 : 성인 어플인 만큼 미성년자의 접근 차단문제가 걸려 있다.

### 특성 유형과 형태

#### 다른 App과의 차별화 1km 나 Hi there~ App 에 없는 증강현실을 이용한 만남주선이 가능하다.



< 그림 9 >

#### 형태 부킹 호프와 유사한 면을 가진 어플리케이션으로, 술친구주선자에 로그인한 이성친구를 찾기위해 증강현실로 찾기, 반경맵으로 찾기, 리스트로 찾기를 제공하고 있다.

#### 맵뷰

< 그림 10 >

사례분석

**사례에 대한 현황**

* ‘1km ‘App은 자신의 위치에서 특정 거리내의 사용자를 검색할 수 있음. 만남 주선은 당사자들의 몫임
* ‘Hi there’ 은 가입된 사용자 모두에게 쪽지를 보낼 수 있음. 하지만 답장회수율이 낮음
* ‘뻐꾸기’ 어플 실행시 자신의 주변에서 가장 가까이 뻐꾸기 어플을 실행하고 있는 상대와 자동으로 1:1 채팅이 시작됨. 대의명분은 채팅임.
* ‘두근두근 우체통’술친구 주선자와 달리 답장이 끊기는 순간 더 이상 상대방에게 연락을 할 수 없음.
* ‘살랑 살랑 돛단배’역시 두근 두근 우체통과 같은 맥락이지만 안드로이드용이라는 점이 다름.
* 기존 유사 기술 적용 분야

- 유사 App -

* 1Km : 아이폰 어플리케이션

|  |  |
| --- | --- |
| mzl  < 그림 11 > | * 자신의 위치를 기초로 불특정 상대방을 검색 * 사진 위주의 정보를 제공 * 프로필 정보를 기초로 상대방을 검색 가능 |

* Hi There : 안드로이드 어플리케이션

|  |  |
| --- | --- |
| 새 비트맵 이미지  < 그림 12 > | * 자신의 위치를 기초로 불특정 상대방을 검색 * 지도에서 타 사용자의 위치 정보를 확인 가능 * 개인 블로그 생성 및 이용 |

* 뻐꾸기 : 안드로이드 어플리케이션

|  |  |
| --- | --- |
| 뻐꾸기  < 그림 13 > | * 상대의 기본 프로필을 확인할 수 있음 * 상위의 x버튼을 누른 뒤 새로고침을 누르면 다시 새로운 상대와 채팅이 시작되는 형식 * 서로의 전화번호도 알 수 없고, 상대가 채팅방을 나가면 더이상 연락할 수 없는 시스템 |

* 두근 두근 우체통 : 아이폰 어플리케이션

|  |  |
| --- | --- |
| 두근두근 우체통  < 그림 14 > | * 익명의 상대에서 문자를 주고 받는 형식. * 문자(엽서)를 보낼 상대의 성별 정도만 선택이 가능하고 랜덤하게 문자를 보내게 됨. * 서로의 연락처를 알 수 없으며 한쪽이 대화를 무시하면 다시는 연락할 수 없는 시스템. |

* 살랑 살랑 돛단배 : 안드로이드 어플리케이션

|  |  |
| --- | --- |
| 살랑 살랑 돛단배 살랑 살랑 돛단배2  < 그림 15 > < 그림 16> | * 익명의 상대에서 문자를 주고 받는 형식. * 문자(엽서)를 보낼 상대의 성별 정도만 선택이 가능하고 랜덤하게 문자를 보내게 됨. * 아이폰의 두근 두근 우체통과 유사함. |

**사례분석 및 실증조사**

### 현황 및 조사대상, 조사방법 등

* 이성을 만나는 여러가지 방법을 웹서핑을 통해 키워드를 얻음.

#### ◎ 소개팅

◎ 모임 (학교/교회/동아리 등)

◎ 인터넷 동호회

◎ 회사

◎ 학원

◎ 도서관

◎ 파티

◎ 헌팅

◎ 나이트클럽 / 힙합클럽

◎ 온라인 미팅 사이트

◎ 결혼정보회사

◎ 온라인채팅

### 사례에 대한 정리

* 소개팅
* 특징 –  
  대한민국의 대표적인 이성을 만나는 방법.  
  이성을 많이 아는 '지인'을 알고 있는 것이 관건.
* 장점 –  
  지인을 통해서 소개받기 때문에 일반적으로 어느 정도 '신뢰감'이 생기게 됨.
* 단점 –  
  간혹 이 신뢰감을 깨뜨리는 소개팅 상대를 만나게 될 경우도 있음.
* 모임 (학교/교회/동아리 등)
* 특징 –  
  평소의 자주 가는 모임에 오빠-동생 (또는 친구, 누나, 선배님..등.) 하다가 연인으로 발전하는 관계.
* 장점 -  
  많은 커플들이 여기서 생겨났고, 이런 만남을 굉장히 선호하는 사람이 많음
* 단점 -  
  모임에 선호하거나 마음에 드는 이성이 있을 확률이 많지 않고, 있다 해도 그녀(또는 그)가 나를 맘에 들어하기도 쉽지 않음.  
  연애로 진전되기에 시간이 상당히 많이 소요되기 쉬움.
* 인터넷 동호회
* 특징 -  
  인터넷 동호회 활동 인구가 많음.  
  와인동호회, 레저 동호회(보드, 인라인 등), 댄스 동호회(살사 등) 등 남 여 비율이 비슷한 수준의 동호회 일수록 이성 친구를 만들 기회가 많이 발생하기도 함.
* 장점 –  
  같은 관심사와 취미를 가진 사람들이 모이는 곳이기 때문에 공감대 형성에 유리함.  
  많은 동호회들이 딱 2-30대 정도의 짝 없는 싱글들이 나오는 경우가 많아서 실제로 연인으로 이어지는 경우도 종종 있음
* 단점 -  
  취미와 관심사보다도 작업질에 혈안이 된 일부 남녀들로 인한 무분별한 물밑 작업들이 난무할 수 있음.  
  실제로 모 동호회에서는 양다리 여자분, 문어다리 작업질 남자 분 등을 몇 번 볼 수 있었음.  
  이런 점이 문제가 되서 일부 동호회는 과다한 작업등으로 인해 문제가 생길 경우 강퇴 조치를 하기도 합니다.
* 회사
* 특징 -  
  바로 사내 연애를 말하는 것임.
* 장점 –  
  서로의 일상에 대한 강력한 공감대 형성이 가능함.  
  연애 상대를 수시로 볼 수 있음.
* 단점 –  
  연인간에 신비감이 떨어질 수 있음  
  회사 내에 연애소식이 퍼질 경우 생길 수 있는 부정적 영향들에 대한 리스크 들이 존재함.
* 학원
* 특징 –  
  뚜렷한 목표가 있는 '고시', '입시' 학원 등 보다는 영어 및 외국어 학원 등이 이성과 마음이 통하기에 다소 수월한 편.
* 장점 -  
  공부하는 모습이 매력적으로 작용할 수 있음

- 단점 –  
공부 본연의 목적에 굉장히 충실한 사람들에게는 접근 자체가 쉽지 않을 수 있음

* 도서관
* 특징 -  
  도서관에 있다고 다 공부하는건 아님.
* 장점 –  
  이성이 많으니 시도해볼 사람이 많음
* 단점 -  
  굉장히 개인적인 공부를 하는 장소이기 때문에 여기서 실행하는 이성에 대한 접근은 거의 '헌팅'과 유사함.
* 파티
* 특징 -  
  클럽에서 종종 진행되는 파티가 아닌 와인파티 또는 남녀 회원이 만나는 소셜파티 등이 이에 해당함.  
  우리나라 파티 문화를 이끌도 지금도 종종 진행되는 곳은 다소 럭셔리 소셜 파티를 표방하는 '클럽프렌즈'가 있음.  
  그외 싸이월드나 네이버 등 인터넷 동호회를 통해서도 파티가 진행됨.
* 장점 -  
  현재는 이러저러한 파티가 많이 생김.
* 단점 –  
  아직도 우리나라에 파티 문화는 대중적이지는 않음.
* 헌팅
* 특징 -  
  선수들의 전유물이라고 여겨짐.
* 장점 –  
  남자 입장에서 볼 때 헌팅의 장점은 외적으로 마음에 드는 이성에게 가장 빠르게 접근 할 수 있는 방법임.
* 단점 –  
  외적인 면 외에는 판단하기 어려운 부분이 있음.  
  거절당했을 경우에 발생하는 굉장한 '민망한&무안함' 이 감당하기에 다소 버거울 수 있음.  
  어느 정도 나이가 들면 헌팅을 하기는 더더욱 어려워기도 함.
* 나이트클럽 / 힙합클럽
* 특징 –  
  하나의 장소에 남자/여자가 모이는 양으로는 최고 수준임  
  굉장히 많은 남녀가 몰리고 굉장히 많은 선수들이 있고, 엄청난 작업들이 오고감.  
  나이트는 '부킹', 클럽은 '부비부비' 시스템을 통해서 이성을 만나게 됨.
* 장점 -  
  멋진 남자, 예쁜 여자들 굉장히 많음.
* 단점 -  
  놀기 위해서 가는 남/녀가 많기 때문에 진지한 연애를 생각하지 않는 사람들이 많음.  
  이성의 내적인 어떤 면을 고려할 시간도 상황도 없음.
* 온라인 미팅 사이트
* 특징 -  
  인터넷이 발달되고 여러가지 온라인 서비스가 생기면서 온라인 미팅 사이트도 많이 생겨남.  
  미국에서 매치닷컴이라는 온라인 미팅 사이트가 인기를 끌면서 우리나라도 유사 서비스가 많이 생김.
* 단점  
  결혼 정도까지의 다소 진지한 분위기의 사이트가 있지만, 굉장히 저급한 사이트도 있음.  
  간혹 프로필 정보를 허위로 작성하는 사람이 있음.
* 결혼정보회사
* 특징 –  
  듀오, 선우 등으로 대표되는 결혼정보 회사가 있음
* 장점 –  
  필요에 정보회사는 많은 경우 굉장히 '정해진 조건'을 중요하게 여겨짐.
* 단점 –  
  많은 비용이 들어감에도 성과가 만족스럽지 못하는 경우가 많음.  
  다소 인위적인 느낌이 들기도 함.
* 온라인채팅
* 특징 –  
  PC 기반의 인터넷 채팅 서비스들 자체가 이용자수가 급감  
  최근 스마트폰 열풍과 함께 하이데어, 후즈히어, 1Km 등의 스마트폰 기반의 온라인 채팅들이 활성화 되고 이를 통해 만남이 생기기도 함.
* 장점 -  
  인터넷 초창기에 온라인 채팅을 통해서 인연을 만나는 분들이 많았음.   
  같은 스마트폰을 가졌다는 동질감과 함께 스마트폰의 위치정보를 확인하여 근거리의 이성에게 접근할 수 있음.
* 단점 -  
  온라인 채팅이 성매매 등 불건전한 만남의 채널로 활용됨.  
  익명성으로 인해 신뢰성이 계속해서 떨어지는 문제가 생김.

프로젝트 수행내용

# 수행범위

### 개발 환경

**Relay Server**

**DataBase**

**Mobile Node**

Oracle XE

TCP/IP (1521)

TCP/IP (18080)

JRE 1.6

<<exeuctionEvironment>>

**ClientAcceptThread**

**DAO**

use

**Naver Map Api**

* 개발 도구
* eclipse-java-helios-SR2-win32 를 기본 개발 언어로 사용한다.
* Java JDK 1.6 을 Java 프로그램 개발 Kit로 사용한다.
* Android SDK 2.1 를 Android 개발 Kit로 사용한다.
* Map 정보 활용을 위해 Naver Map Api를 사용한다.
* http://dev.naver.com/openapi/apis/map/android/example
* Motorola Motoroi (Android version: 2.1)을 기본 테스트기로 사용한다.
* Oracle 10g XE 를 Database로, SQL developer를 Database 관리 도구로 사용한다.
* Git(eGit 1.0.0) 을 형상 관리 툴로서 사용한다.
* http://www.eclipse.org/egit/
* 그 외 설계를 위해 StarUML, ERWin 등 을 사용한다.

### 개요

[시스템 개요]

- 필수 기능

* 회원 가입 및 로그인
* 게시 정보를 입력
* 메시지 전달 및 수신
* 위치 정보를 통한 지도 표현
* 위치 정보를 바탕으로 증강 현실 구현

- 사용 개요

* 1. 만남을 원하는 사용자가 관련 정보를 게시한다.
  2. 자신의 위치 정보를 바탕으로 하여, 거리가 가까운 순서로 만남을 원하는 사용자가 게시한 정보를 리스트 형태로 게시한다.
     + Map, 증강현실 형태의 정보 보기로 전환가능 하다.
  3. 게시된 정보를 통해 마음에 드는 상대에게 메시지를 보낸다.
  4. 메시지를 받은 이용자는 상대방의 정보를 확인한 후, 허락 여부를 결정하여 연락을 취한다.

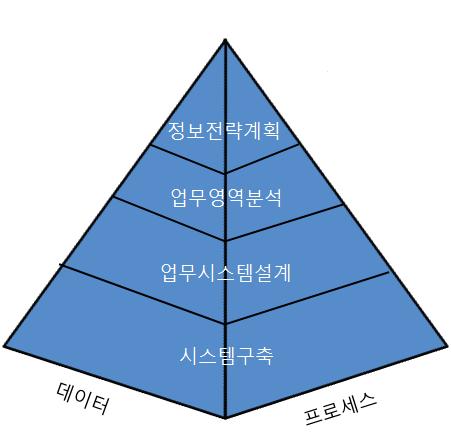
[뷰 개요]

|  |  |
| --- | --- |
| 리스트 | * 게시정보: 자신 및 일행을 홍보 * 게시정보를 통해 게시자의 프로필을 확인 가능 |
| **MAP**  **검색조건 리스트 증강현실** | * 지도에서 타 사용자의 위치 확인 * 상점 등 주변 정보 확인 |
| 증강현실 | * 카메라 모드를 기초로 타 사용자의 방향과 거리를 표현 |

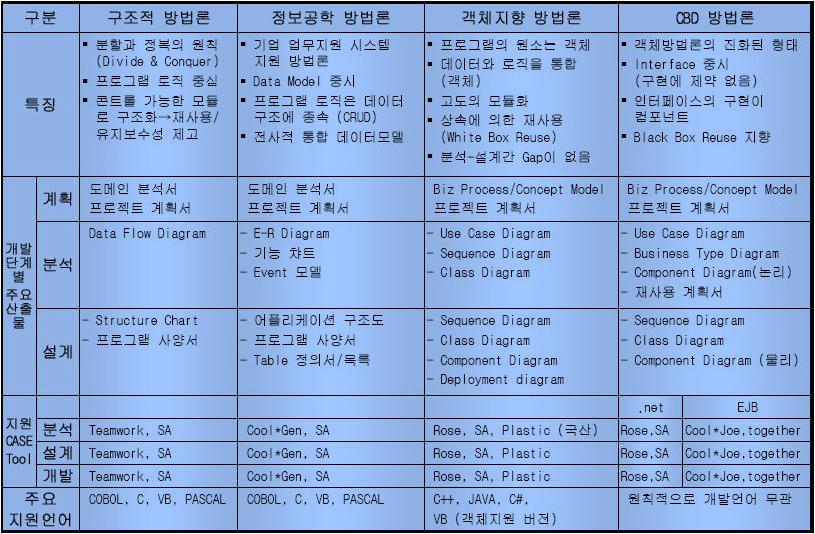
[모델 개요]

* 개인 정보
  + 필수 기재 사항: 아이디, 암호, 성별
  + 선택 기재 사항: 이름, 나이, 사진, 홈페이지, 주소, 전화번호
* 게시 정보
  + 필수 기재 사항 : 사진, 인원, 위치
  + 선택 기재 사항 : 나이, 주량, 코멘트

### DB 모델링



< 그림 17 > 기본적인 모델링 구축은 위의 그림과 같다.

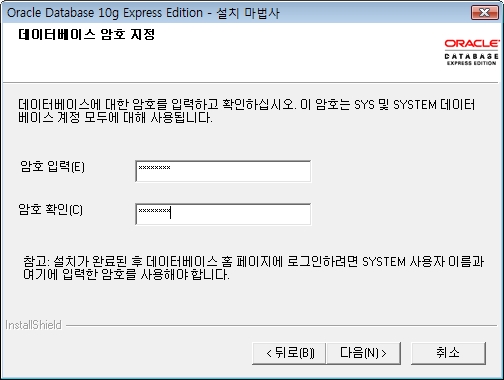


< 그림 18 > 여러가지의 개발방법중에 객체지향 방법론을 사용하였다.

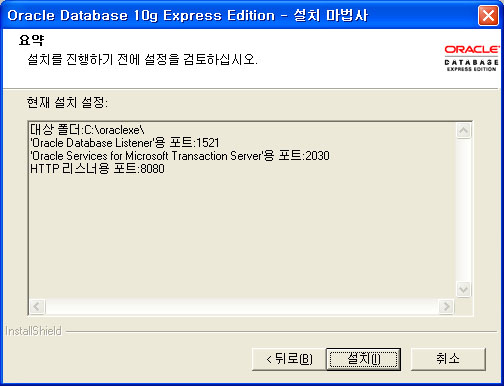
### 설치법

* 설치 방법

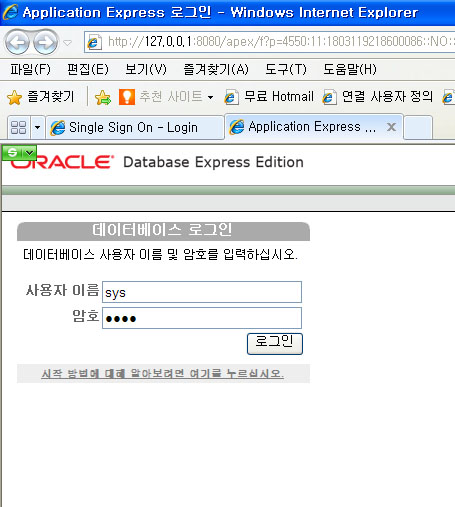
1. 데이터베이스
   1. 데이터베이스 설치 방법
      1. 데이터베이스는 Oracle 10g Express Edition 버전을 사용한다.
         1. 아래 사이트에서 해당 운영체제에 알맞은 버전을 찾아 설치한다.
            1. <http://www.oracle.com/technetwork/database/express-edition/downloads/index.html?ssSourceSiteId=ocomen>
      2. 암호를 입력하고, 기록해 둔다



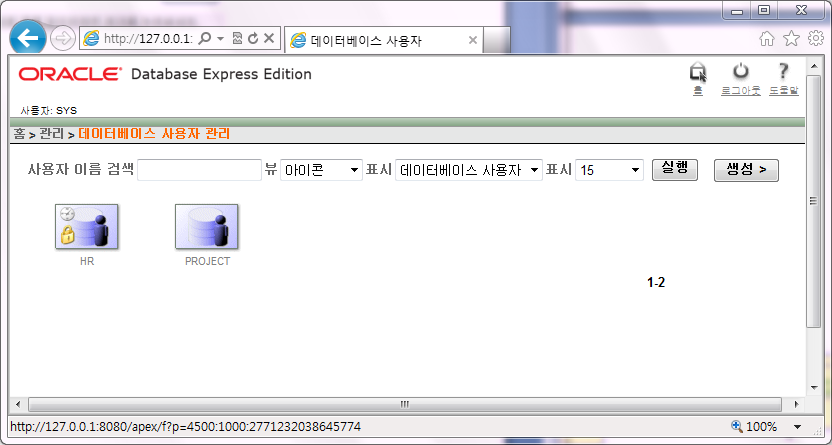
* + 1. 설치 시, 사용 포트는 1521 포트를 사용한다



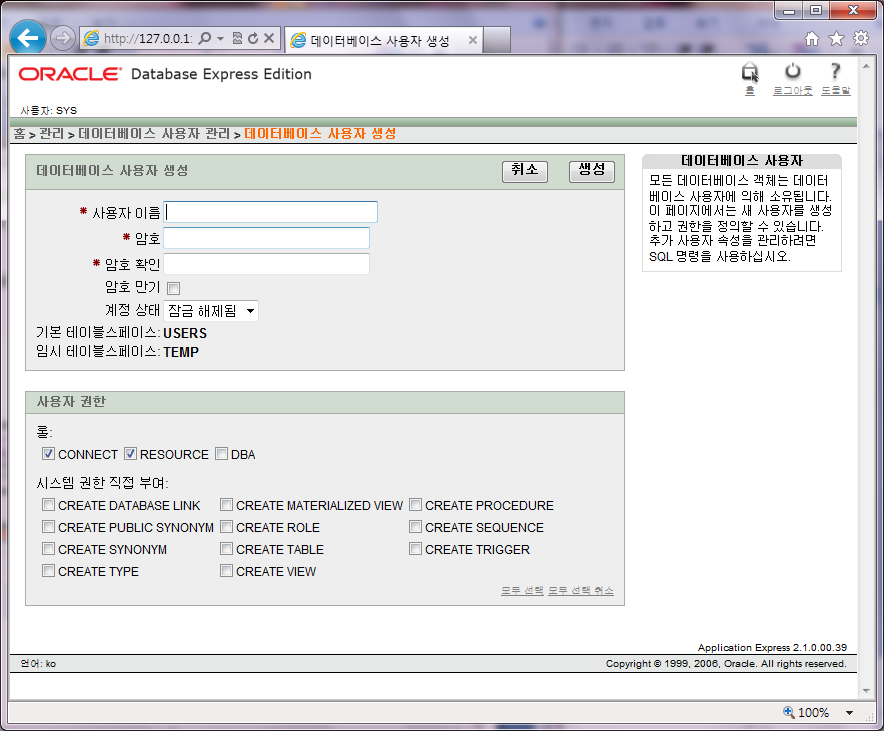
* 1. 데이터베이스 User 생성
     1. 데이터베이스 로그인
        1. 사용자 이름: sys
        2. 암호: 설치시 등록한 암



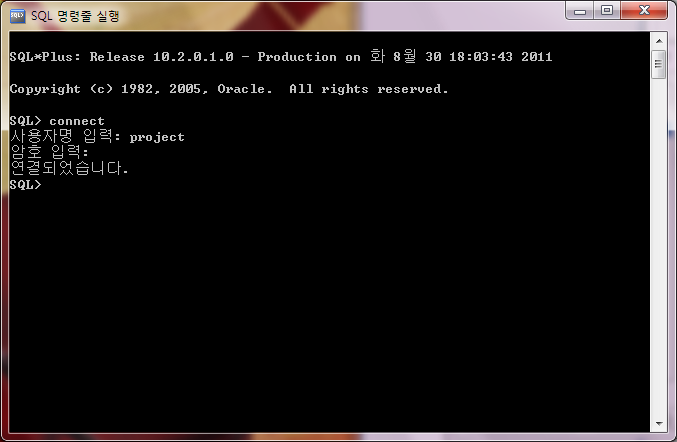
* + 1. 홈 > 관리 > 데이터베이스 사용자 관리 페이지로 들어간다.



* + 1. 생성 페이지에서 사용자를 등록한다.
       1. 사용자 이름 : Project
       2. 암호 : kitri
       3. 필수 사용자 권한
          1. CREATE TABLE
          2. CREATE SEQUENCE



* 1. SQL Query 입력
     1. SQL 명령줄 실행



* + 1. 데이터베이스 로그인 후, [데이터베이스 명세]에 첨부된 SQL Query를 모두 입력한다.

1. Java
   1. Java JDK 설치
      1. Java JDK 는 JAVA SE 6 버전을 사용한다.
      2. 자신의 운영체제에 알맞은 버전을 아래의 사이트에서 설치한다
         1. <http://www.oracle.com/technetwork/java/javase/downloads/jdk-6u27-download-440405.html>
2. Android
   1. Android APK 설치
   2. Mobile Service.apk 파일을 테스트 용 기기에 설치한다.
      1. 안드로이드 테스트 기기 요구조건
         1. 안드로이드 2.2 버전 이상의 커널
         2. SD 카드
         3. Orientation 센서
         4. GPS 센서
         5. 네트워크 사용
      2. 설치 파일 URI :
         1. <https://github.com/tinywind/HelloDrinking/blob/master/Reference%20and%20Document/HelloDrinking.apk>
3. 프로그램 실행
   1. 안드로이드
      1. 설치된 프로그램 중 [Mobile Service] 를 실행한다.
   2. 서버
      1. 서버 프로그램의 실행가능한 jar 파일을 다운받는다.
         1. <https://github.com/tinywind/HelloDrinking/blob/master/Reference%20and%20Document/ServiceServer.jar>
      2. 다음 명령을 실행한다.
         1. Java –jar ServiceServer.jar

### 시스템 아키텍처 모델

#### Business Model

사용자

사용자

술친구  
주선자

광고주

* + 사용자에게 매칭 서비스를 제공한다.
  + 매칭을 지원하기 위한 장소 등의 기타 정보를 제공한다.
  + 수집된 사용자 정보를 바탕으로 기업의 광고 등 마케팅 활동을 지원한다.
  + 광고 효율을 바탕으로 광고주에게서 광고 수수료를 지불 받는다.

# Achitecture Model

메시지 수신/발신

매칭 및 장소   
정보 제공 서비스

사용자 정보 수집

광고 수수료

마케팅 활동 지원

Client

Relay Server

<<실행 환경>>

Android 2.1

# 개발 환경

### 개발툴

* + eclipse-java-helios-SR2-win32 를 기본 개발 언어로 사용한다.
  + Java JDK 1.6 을 Java 프로그램 개발 Kit로 사용한다.
  + Android SDK 2.1 를 Android 개발 Kit로 사용한다.
  + Map 정보 활용을 위해 Naver Map Api를 사용한다.

* + - <http://dev.naver.com/openapi/apis/map/android/example>
  + Motorola Motoroi (Android version: 2.1)을 기본 테스트기로 사용한다.
  + Oracle 10g XE 를 Database로, SQL developer를 Database 관리 도구로 사용한다.
  + Git(eGit 1.0.0) 을 형상 관리 툴로서 사용한다.

* + - <http://www.eclipse.org/egit/>
  + 그 외 설계를 위해 StarUML, ERWin 등을 사용한다.

### 버전관리계획

* + 개발자 로컬 PC의 Android Emulator 환경을 기반으로 개발하는 것을 기본으로 하며, 타 개발자 작업 내용의 참조가 요구되는 경우에는 해당 작업 내용을 개발자 로컬 PC로 이전받아 참조하며, 공용 라이브러리의 확장은 최종 개발 책임자에게 제작을 의뢰한다.
  + Data Base 서버의 관리 및 작업 내용 반영은 해당 개발 책임자가 작업한다.
  + Relay Server 의 관리 및 작업 내용 반영은 해당 개발 책임자가 작업한다.
  + 단위 개발 작업이 완료되면, 최종 개발 책임자는 해당 결과물을 받아 테스트 서버에 설치한다.
  + 개발 진행중인 작업 내용의 경우 개인 단위의 백업관리가 이루어지며, 개발 완료 작업 내용과 일(daily)단위의 백업은 최종 개발 책임자가 맡는다.

### 데이터베이스 접근 계획

* + 모든 개발자는 하나의 데이터베이스 서버(테스트 혹은 운영서버)를 공유하여 사용한다.
  + 데이터베이스의 접근은 Relay Server의 JDBC를 이용하여 접근하며, 자체 Database Pooling모듈을 사용하며, 필요한 경우 DB 리소스는 독립적으로 획득할 수 있다.

# 명명 규칙

* + DB / Relay Server / Mobile Service 의 명명은 각기 해당 책임자가 세부 규칙을 확정하며, 모두 별개의 명명 규칙을 가진다.
  + DB 의 테이블 및 테이블의 필드를 포함하는 내부 스키마는 명명규칙은 해당 책임자가 개별 작성한다.
  + Relay Server의 명명규칙은 Mobile Service의 명명 규칙과 공유한다.
  + Mobile Service의 리소스 자원의 파일명은 리소스 id에 관련 파일 형식의 확장자를 덧붙인다.
  + Relay Server 의 리소스 자원의 파일명은 Mobile Service의 리소스 id에 관련 파일 형식의 확장자를 덧붙인다..

### 디렉토리 명명 규칙

* + 영문 혹은 숫자로 구성하되 소문자를 사용하며, 첫글자는 반드시 영문자로 한다.
  + 특수문자의 사용은 공백문자를 대신하여 밑줄(\_)을 사용하며, 그 외 특수문자 사용을 불허한다.

### Class 및 interface 명명 규칙

* + 영문 혹은 숫자로 구성하며, 모든 특수문자의 사용을 불허한다.
  + 첫 글자는 영문 대문자로 한다.
  + 공백을 대신하여 공백 이후 나타나는 첫 글자를 대문자로 표현한다.
  + Interface는 형용사를 사용해, 구현 클래스가 하는 일을 명확히 알 수 있도록 한다.
  + DAO, DTO 는 해당 사용 테이블 또는 뷰의 이름의 뒤에 DAO, DTO를 덧붙인다.
  + 구현 클래스의 경우, 객체를 만들어 사용하는 모든 클래스는 명사로 지정하며, 그 외 행동만을 가지고 있는 클래스(맴버 변수를 가지지 않는 클래스)는 동사를 사용하여 명명한다.
  + Mobile Service의 뷰를 표현하는 Activity 및 Dialog클래스의 명명규칙은 다음과 같다.
    - Activity 를 상속받아 사용하거나, 클래스의 주 역할이 직접적인 Activity 생성 및 사용에 있다면, 이름의 마지막에 Activity를 덧붙인다.
    - Activity 클래스 이름의 처음은 나타내고자 하는 데이터 모델 및 배경을 나타낼 수 있는 명사를 사용한다.
    - 해당 클래스의 주목적에 해당하는 동사를 ‘Activity’ 앞에 삽입한다. 하지만 주목적이 데이터 모델를 보여주는데 있다면 동사를 생략한다.
      * 예) MemberJoinActivity, MemberDropActivity, ProfileActivity, MapActivity
    - 만약 Activity의 스타일이 해당 Activity를 호출하는 Activity 내부에 표현되는 Dialog 스타일이라면 이름의 마지막에 Activity를 Dialog로 치환한다.

### Class 내부의 변수 및 메서드 명명 규칙

* + 메서드는 첫단어를 역할에 관련한 동사로 하며, 그후 단어는 역할에 대한 목적어를 사용하며, 여러 역할을 동시에 하는 메서드는 And 로 그 역할을 구분한다.
    - Examples : sendQuery, sendQueryAndGetResult
  + 변수는 첫 단어를 해당 변수의 데이터타입을 표현하며, 그 이후 변수의 역할 및 성질에 관련된 단어를 덧붙인다.
  + 클래스의 맴버 변수의 경우 첫 글자를 m 으로 시작하며, 이후는 변수의 명명규칙을 동일하게 따른다.
    - Examples : mIntNumOfLecturerooms, mLinearIpHistory
  + 상수는 모두 대문자로 표현한다.
  + 변수의 경우 단순 증감이며 for문 등의 제어문 내에서 선언되는 경우 <i,j,k> 등의 간단한 소문자를 사용한다.(‘p’는 예약된 변수로써, PreparedStatement의 index에서 사용한다.
    - 예) for(int i=0;i<10;++i)

### 리소스 id의 명명 규칙

* + 영문 소문자와 숫자, 밑줄(\_)만을 사용한다.
  + 첫 글자는 반드시 영문 소문자로 한다.
  + Activity 내부에서 사용되는 리소스 id의 명명 규칙
    - 사용되는 Activity 클래스명으로 시작하며 Activity 클래스의 Activity 및 Dialog는 제거한다. 그리고 대문자는 모두 소문자로 치환한다. 그리고 밑줄(\_)을 덧붙인후 리소스의 종류를 명시하고, 다시 밑줄(\_)을 덧붙인후 리소스의 역할에 관련한 단어를 덧붙인다.
      * Examples : ipmanagement\_button\_send
    - Activity의 Layout 리소스의 명명은 Actvity 클래스 명에서 Actvity를 제외한 이름으로 한다.
      * Examples : memberjoin.xml
    - 여러 Activity에서 공동적으로 사용되는 리소스는 앞의 명명규칙에서 Activity 이름과 첫 밑줄(\_)을 제거한다.
      * Examples : style\_custombutton
  + 사용되는 리소스의 확장자는 아래와 같다.

|  |  |
| --- | --- |
| 파일 형식 | 파일 확장자 |
| image | .png, .bmp, .jpg |
| sound | .wav, .mid |
| Config | .properties |
| Xml | .xml |

표 4 파일 확장자

### 데이터베이스 명명 규칙

* + 테이블 명칭 : 정보의 성격을 나타낼 수 있는 명사를 사용한다.
    - 예) EMPLOYEE(직원), COURSE(과정)
  + 필드 명칭 : 테이블마다 공통적인 id와 name필드의 경우 테이블 명칭의 약자에 구분자(‘\_’)를 붙이는 형태로 사용하며, 테이블의 고유한 필드인 경우 데이터의 성격을 나타낼 수 있는 명사를 사용한다.
    - 예1) emp\_id, std\_id
    - 예2) lc\_name, emp\_name, std\_name
  + Table Constraint : 각 테이블의 제약조건은 아래와 같이 명명한다.
    - Primary\_Key : 기본키는 해당 테이블의 약어와 PK\_ 를 같이 사용한다.
      * 예: PK\_USERINFO\_ID
    - Foreign\_Key : 외뢰키를 설정할 두 테이블의 약어와 FK\_ 를 같이 사용한다.
      * 예: FK\_POSTINFO\_ID
    - Check : 체크할 테이블의 약어와 컬럼명, CH\_ 를 같이 사용한다.
      * 예: CH\_USERTINFO\_SEX
  + 트리거 명칭 : 해당 테이블 명에 구분자(‘\_’)를 붙인 후 트리거의 역할을 나타내는 명사를 사용한다.
    - 예) ip\_log
  + Index 명칭 : INDEX\_ 에 해당 테이블명, 구분자(‘\_’), 해당 열의 이름을 붙인다.
    - 예) INDEX\_USERINFO\_ID

# 개발 구조

### 디렉토리 구조

* + 경로는 Document Root 을 기준
  + Directory 구조
    - Java Source 디렉토리 : \src
    - Java Class 디렉토리 : \bin
    - 리소스 디렉토리 : \res
    - 설정 디렉토리 : \properties
    - 기타 라이브러리 : \lib
  + Mobile Service 의 Resource 파일 디렉토리 : Android 개발 표준을 따름
    - Layout 파일 : \res\layout
    - 이미지 파일 디렉토리 : \res\drawable-hdpi
    - 애니메이션 파일 디렉토리: \res\anim
    - 사운드 파일 디렉토리 : \res\raw

### 공용 라이브러리 구조

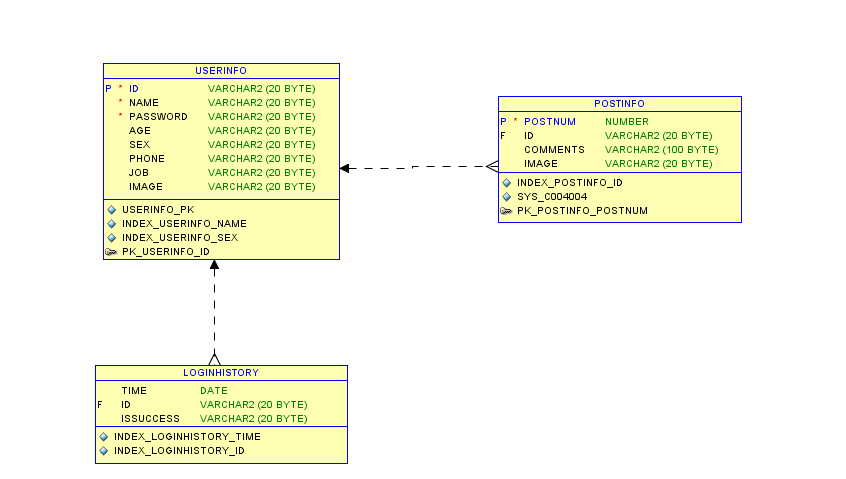
* + 서비스 독립적으로 사용될 수 있는 라이브러리들의 집합
  + Relay Server 와 Mobile Service 간의 공통으로 사용되는 라이브러리
    - 통신 패키지: .class 파일과 .properties 파일을 포함한다.
    - DAO 및 DTO class 의 정보를 가진다.

# 기능 요구서

| 회원 관리   1. 로그인    1. 로그인기능       1. 아이디 : 아이디를 입력한다.       2. 비밀번호 : 비밀번호를 입력한다.    2. 회원가입기능       1. 아이디가 없을 경우 회원가입을 할 수 있는 페이지가 있어야 한다. 2. 회원 가입    1. 필수 기재 사항       1. 아이디 : 영문, 숫자       2. 암호 : 4자이상       3. 성별 : 라디오 버튼    2. 선택 기재 사항       1. 이름       2. 나이       3. 사진 : 즉석촬영 또는 기존에 저장된 사진을 사용할 수 있어야 한다.       4. 홈페이지       5. 주소       6. 전화번호 3. 회원 정보 수정    1. 변경 불가 정보       1. 아이디 : 영문, 숫자       2. 성별 : 라디오 버튼       3. 이름 : 개명시 관리자를 통해 변경 가능       4. 나이 : 매년마다 1씩 증가한다.    2. 변경 가능 정보       1. 암호 : 4자이상       2. 사진 : 즉석촬영 또는 기존에 저장된 사진을 사용할 수 있어야 한다.       3. 홈페이지       4. 주소       5. 전화번호 | |
| --- | --- |
| 사용자 매칭   1. 증강현실    1. 나침반       1. 현재 자신이 바라보고 있는 방위가 나타난다.    2. 썸네일 사진       1. 카메라에 비춰지는 방향에 있는 이성의 썸네일 사진이 나타난다.       2. 썸네일 사진을 클릭하면 간단 프로필 정보가 나타난다.       3. 간단 프로필 화면에서 쪽지 보내기를 할 수 있다.       4. 간단 프로필 화면에서 상세 프로필로 넘어갈 수 있다. 2. Map    1. 나침반       1. 현재 자신이 바라보고 있는 방위를 알 수 있다.    2. 썸네일 사진       1. 일정 반경 내에 있는 이성의 썸네일 사진이 나타난다.       2. 썸네일 사진을 클릭하면 간단 프로필 정보가 나타난다.       3. 간단 프로필 화면에서 쪽지 보내기를 할 수 있다.       4. 간단 프로필 화면에서 상세 프로필로 넘어갈 수 있다. 3. 리스트    1. 리스트 화면       1. 간단 프로필이 리스트로 나열되어진 형태이다.       2. 여러명을 찜할 수 있다.       3. 쪽지 보내기 버튼이 있다.       4. 프로필 화면에서 상세정보로 넘어갈 수 있다. |
| 프로필 관리   1. 간단 프로필    1. 상대방의 프로필, 인원, 코멘트를 볼 수 있다.    2. 프로필은 아이디, 나이, 썸네일 사진으로 되어 있다.    3. 상세 프로필 보기 버튼이 있다.    4. 쪽지 보내기 버튼이 있다. 2. 상세 프로필    1. 사용자가 모든 회원 정보가 공개 된다.       1. 아이디       2. 성별       3. 이름       4. 나이       5. 사진       6. 홈페이지       7. 주소       8. 전화번호    2. 비공개 여부를 설정할 수 있다. | |

# 데이터베이스

DB User ID : “PROJECT”



### USERINFO : 가입된 사용자의 정보 테이블

|  |  |  |
| --- | --- | --- |
| Column ID | Column Name | Description |
| 1 | ID | 고유한 식별자 |
| 2 | NAME | 사용자 이름 |
| 3 | PASSWORD | 로그인시 사용되는 암호 |
| 4 | AGE | 사용자 나이 |
| 5 | SEX | 사용자 성별 (도메인: ‘MAN’ or ‘WOMEN’ |
| 6 | PHONE | 사용자 휴대폰 번호 |
| 7 | JOB | 사용자 직업 |
| 8 | IMAGE | 사용자를 담고 있는 그림 파일의 경로 |

SQL Query :

CREATE TABLE "PROJECT"."USERINFO"

(

"ID" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"NAME" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"PASSWORD" VARCHAR2(20 BYTE) NOT NULL ENABLE,

"AGE" VARCHAR2(20 BYTE),

"SEX" VARCHAR2(20 BYTE),

"PHONE" VARCHAR2(20 BYTE),

"JOB" VARCHAR2(20 BYTE),

"IMAGE" VARCHAR2(20 BYTE),

CONSTRAINT

"PK\_USERINFO\_ID" PRIMARY KEY ("ID")   
USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT) TABLESPACE "USERS" ENABLE

)

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

CREATE INDEX "PROJECT"."INDEX\_USERINFO\_NAME" ON "PROJECT"."USERINFO"

(

"NAME"

)

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

CREATE INDEX "PROJECT"."INDEX\_USERINFO\_SEX" ON "PROJECT"."USERINFO"

(

"SEX"

)

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

CREATE UNIQUE INDEX "PROJECT"."USERINFO\_PK" ON "PROJECT"."USERINFO"

(

"ID"

)

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

### POSTINFO : 게시물의 정보 테이블

|  |  |  |
| --- | --- | --- |
| Column ID | Column Name | Description |
| 1 | POSTNUM | 고유한 식별자 |
| 2 | ID | 사용자 식별자 |
| 3 | COMMENT | 전달하는 내용 |
| 4 | IMAGE | 해당 Post에 대한 내용을 담고 있는 그림 파일의 경로 |
| 5 | LONGITUDE | 해당 Post를 올릴 당시의 경도 좌표 |
| 6 | LATITUDE | 해당 Post를 올릴 당시의 위도 좌표 |

SQL Query:

CREATE TABLE "PROJECT"."POSTINFO"

(

"POSTNUM" NUMBER,

"ID" VARCHAR2(30 BYTE),

"COMMENT" VARCHAR2(100 BYTE),

"IMAGE" VARCHAR2(100 BYTE),

"LONGITUDE" NUMBER,

"LATITUDE" NUMBER,

CONSTRAINT "PK\_POSTINFO\_POSTNUM" PRIMARY KEY ("POSTNUM") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT) TABLESPACE "USERS" ENABLE,

CONSTRAINT "FK\_POSTINFO\_ID" FOREIGN KEY ("ID") REFERENCES "PROJECT"."USERINFO" ("ID") ENABLE

)

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

CREATE INDEX "PROJECT"."INDEX\_POSTINFO\_ID" ON "PROJECT"."POSTINFO"

(

"ID"

)

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

CREATE UNIQUE INDEX "PROJECT"."SYS\_C004004" ON "PROJECT"."POSTINFO"

(

"POSTNUM"

)

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

### LOGINHISTORY : 로그인 시도 기록 테이블

|  |  |  |
| --- | --- | --- |
| Column ID | Column Name | Description |
| 1 | TIME | 로그인을 시도한 시각 |
| 2 | ID | 사용자 식별자 |
| 3 | ISSUCCESS | 로그인 성공 여부 |

SQL Query :

CREATE TABLE "PROJECT"."LOGINHISTORY"

(

"TIME" DATE,

"ID" VARCHAR2(20 BYTE),

"ISSUCCESS" VARCHAR2(20 BYTE),

CONSTRAINT "FK\_LOGINHISTORY\_ID" FOREIGN KEY ("ID") REFERENCES "PROJECT"."USERINFO" ("ID") ENABLE

)

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

CREATE INDEX "PROJECT"."INDEX\_LOGINHISTORY\_ID" ON "PROJECT"."LOGINHISTORY"

(

"ID"

)

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

CREATE INDEX "PROJECT"."INDEX\_LOGINHISTORY\_TIME" ON "PROJECT"."LOGINHISTORY"

(

"TIME"

)

PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS STORAGE

(

INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER\_POOL DEFAULT

)

TABLESPACE "USERS" ;

### 사용 Sequence

|  |  |
| --- | --- |
| Sequence ID | Description |
| POST\_SEQ | POSTINFO 테이블의 POSTNUM 레코드 삽입을 위한 Squence |

CREATE SEQUENCE "PROJECT"."POST\_SEQ" MINVALUE 1 MAXVALUE 100000 INCREMENT BY 1 START WITH 21 CACHE 20 NOORDER NOCYCLE ;

# Mobile Service

### 디렉토리 구성

|  |  |
| --- | --- |
| **Directory Name** | **Description** |
| **src** | \*.java Files |
| **gen** | Auto Generated Java Files |
| **lib** | External Library |
| **bin** | \*.class Files |
| **res** | Resource Files |

### 패키지 구성

|  |  |
| --- | --- |
| **Packege Name** | **Description** |
| **com.nhn.android.maps** | Naver Map Api Toturial Packege |
| **kr.android.hellodrinking** | Application 작성 및 Application 과 Application Context와 통신에 사용하는 클래스의 작성에 필요한 클래스를 제공합니다. |
| **kr.android.hellodrinking.activitiy** | 사용자 인터페이스의 중 Activity 의 작성 및 Activity의 생성에 필요한 클래스를 제공합니다. |
| **kr.android.hellodrinking.dialog** | 사용자 인터페이스의 중 Dialog 의 작성 및 Dialog 의 생성에 필요한 클래스를 제공합니다. |
| **kr.android.hellodrinking.sensor** | 사용되는 Sensor 및 SensorManager 에 대해서 재정의한 클래스를 제공합니다. |
| **kr.android.hellodrinking.utillity** | 수치 연산 및 그래픽 연산, 파일 입출력에 관련한 클래스를 제공합니다. |
| **kr.android.hellodrinking.view** | 사용자 인터페이스 중 View 의 작성 및 View의 생성에 필요한 클래스를 제공합니다. |
| **kr.android.hellodrinking.transmission** | 네트워크 어플리케이션의 구현을 위한 클래스를 제공합니다. |
| **kr.android.hellodrinking.transmission.dto** | Bean 의 생성에 필요한 클래스를 제공합니다. |
| **kr.android.hellodrinking.transmission.exception** | 통신 중 일어날 수 있는 예외에 대해 정의한 클래스를 제공합니다. |

클래스 관계 개요도

<<실행 환경>>

JRE 1.6

Database Server

<<실행 환경>>

Oracle 10g

<<실행 환경>>

Naver Map Api

TCP/IP

TCP/IP

Application

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | **해당 Class** |
| **Application** | 글로벌 변수 및 상태 정보를 가지고 있는 기초 클래스 | HelloDrinkingApplication |
| **Activity / View** | 데이터 모델을 뷰에 표현하기 위한 컨트롤을 담당하는 클래스  \*뷰에 관련한 정보는 .xml 파일이 담고 있다. | \*Activity, \*Dialog, \*View :  ARActivity, FrameActivity, LoginActivity, MapActivity, MemberinfoActivity, MemberJoinActivity, NMapFrameActivity, PostActivity, PostsActivity, ConfigureDialog, UserInfoDialog, CameraView, CompassView, MapContainerView |
| **Utils** | Activity 또는 View에서 사용하기 위한 연산을 담당하는 클래스 | Calculations, GraphicUtils |
| **Sensor** | SensorManager 의 역할을 확장하는 클래스 | Compass |
| **Data Model** | Activity 또는 View 에 표현되는 정보를 담고 있는 클래스  \* 서버와 동일한 클래스가 사용된다. | Bean, PostBean, UserBean |
| **Request** | 메서드를 호출하기 위해서 필요한 정보를 포함한 클래스  \* 서버와 동일한 클래스가 사용된다. | Request, RequestBeanPackege |
| **Responce** | Request를 받아 처리된 정보를 포함한 클래스  \* 서버와 동일한 클래스가 사용된다. | ResponceBeanPackege |

Application Classes

Activity / View

Sensor / Utils /   
Graphic Components

**MAP**

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| **HelloDrinkingApplication** | PostBean 의 정보 및, 로그인 된 사용자의 상태 정보를 가지고 있는 기초 클래스 |

class HelloDrinkingApplication extends Application

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **HelloDrinkingApplication()** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ setServerFromPreferences()** | void | Preferences 객체에서 mServerIp, mServerPort 필드를 초기화한다. |
| **+ addPost(PostBean)** | void | 지정 PostBean을 등록한다. |
| **+ removePost(PostBean)** | void | 지정 PostBean을 삭제한다. |
| **+ getListPosts()** | List<PostBean> | 저장된 PostBean 객체들을 List 형태로 반환한다. |
| **+ setListPosts(List<PostBean>)** | void | 지정 List의 PostBean 객체들로 저장된 PostBean 객체들을 대체한다. |
| **+ setId(String)** | void | 로그인된 User의 Id를 저장한다. |
| **+ getId()** | String | 로그인된 User의 Id를 반환한다. |
| **+ refreshListPosts(NGeoPoint, int)** | void | 지정 NGeoPoint 객체의 좌표로부터 지정한 거리 내의 PostBean 객체를 서버로부터 가져온다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **+ (s) (f) DEFAULT\_SEARCH\_DISTANCE** | int | 거리 조건 입력에 사용되는 기본 거리 : 10000 |
| **+ (s) mServerIp** | String | 서버의 Ip를 가르킨다. |
| **+ (s) mServerPort** | int | 서버의 Port를 가르킨다. |
| **- id** | String | 로그인 된 사용자의 Id를 가르킨다. |
| **- mListPosts** | List<PostBean> | 게시물 객체를 가르킨다. |

Activity / View Classes

Data Model

서버

Responce

Request

HelloDrinkingApplication

PostBean

UserBean

154

MapContainerView

FrameActivity

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| **ARActivity** | 증강현실 Activity |
| **FrameActivity** | 레이아웃 및 동일 기능을 정의하는 추상 Activity |
| **LoginActivity** | 로그인 Activity |
| **MapActivity** | 지도 형태 Activity |
| **MemberinfoActivity** | 로그인된 회원의 정보 확인 및 수정을 위한 뷰의 Activity |
| **MemberJoinActivity** | 회원 가입을 위한 Activity |
| **NMapFrameActivity** | 레이아웃 및 동일 기능을 정의하는 추상 NMapActivity   * FrameActivity와 동일한 내용을 담고 있다. |
| **PostActivity** | 게시물 등록 Activity |
| **PostsActivity** | List 형태를 지닌 Activity |
| **ConfigureDialog** | Server의 Ip 및 Port 를 설정하는 Dialog |
| **PostInfoDialog** | 선택된 게시물의 상세 정보를 표현하는 Dialog |
| **CameraView** | 카메라에서 받아온 이미지를 표현하는 View |
| **CompassView** | 나침반의 이미지를 표현하는 View |
| **MapContainerView** | 지도의 이미지를 표현하는 View |

class ARActivity extends FrameActivity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **= loadContent()** | void | ar.xml의 내용을 표현한다. 그리고 presentPosts()를 실행한다. |
| **+ onAccuracyChanged(Sensor, int)** | void | Sensor의 정확도가 변경될 때 호출된다. 실제로는 어떠한 일도 하지 않는다. |
| **= onResume()** | void | Activity가 Resume 상태가 될 때 호출된다. Compass 객체의 리스너를 등록한다. |
| **+ onSensorChanged(SensorEvent)** | void | Sensor의 값이 변경될 때 호출된다. Post의 위치를 계산하여 화면에 표현한다. |
| **= onStop()** | void | Activity가 Stop 상태가 될 때 호출된다. Compass 객체의 리스너를 해지한다. |
| **= presentLocationChanged()** | void | 자신의 위치가 변경될 때 호출된다. 실제로는 어떠한 일도 하지 않는다. |
| **= presentPosts()** | void | Application에 등록된 PostBean 객체의 정보를 바탕으로 View를 만든다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **+ (s) (f) HORIZONAL\_VIEWING\_ANGLE** | double | 화면에 표현되는 수평 시야각 : 30도 |
| **+ (s) (f) VERTICAL\_VIEWING\_ANGLE** | double | 화면에 표현되는 수직 시야각 : 60도 |
| **- (s) (f) STANDARD\_COORDINATE** | doauble[] | 화면에 표현되는 좌표의 범위를 구하기 위한 단위 좌표 : (0,0,1) |
| **- (s) (f) LEFT\_ON\_SCREEN** | double | 수평 시야각으로부터 화면에 표현되는 최초의 왼쪽 좌표를 구한다.   * 화면 중앙을 0이라 칭한다. * 기준되는 계산에 이용되는 단위 좌표는 STANDARD\_COORDINATE 이다. |
| **- (s) (f) UP\_ON\_SCREEN** | double | 수직 시야각으로부터 화면에 표현되는 최초의 윗쪽 좌표를 구한다.   * 화면 중앙을 0이라 칭한다. * 기준되는 계산에 이용되는 단위 좌표는 STANDARD\_COORDINATE 이다. |
| **- mCompass** | Compass | Listener 등록 및 해지를 위한 객체 |
| **- mCompassView** | CompassView | 나침반을 표현하는 객체 |

abstract class FrameActivity extends Activity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | frame.xml의 내용을 표현한다. 그리고 loadContent()를 실행한다. |
| **= onResume()** | void | Activity가 Resume 상태가 될 때 호출된다. startMyLocation()을 실행한다. |
| **= onStop()** | void | Activity가 Stop 상태가 될 때 호출된다. stopMyLocation()을 실행한다. |
| **= startMyLocation()** | void | mMapLocationManager의 위치 탐색을 시작한다. 만약 사용이 불가능한 상태라면, GPS 설정 화면으로 넘어간다. |
| **= stopMyLocation()** | void | 위치 탐색을 종료한다. |
| **+ onLocationChanged( NMapLocationManager, NGeoPoint)** | void | myLocatioin을 갱신한다. 그리고 presentLocationChanged()를 실핸한다. |
| **+ onLocationUpdateTimeout( NMapLocationManager)** | void | stopMyLocation()을 실행한다. 그리고 에러 메세지를 출력한다. |
| **= abstract loadContent()** | void | 페이지가 만들어질 때 실행된다. |
| **= abstract presentPosts()** | void | 게시물을 표현한다. |
| **= abstract presentLocationChanged()** | Void | 위치가 변할 때 호출된다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **= mInflater** | LayoutInFlater | Layout xml 파일을 뷰로 가져오기 위한 객체 |
| **= mViewgroup** | ViewGroup | 상속받은 클래스의 뷰를 담기 위한 객체 |
| **= mButtonPosts** | ImageButton | PostsActivity 로 인텐트시키는 객체 |
| **= mButtonMap** | ImageButton | MapActivity 로 인텐트시키는 객체 |
| **= mButtonAR** | ImageButton | ARActivity 로 인텐트시키는 객체 |
| **= mButtonMember** | ImageButton | MemberinfoActivity 로 인텐트시키는 객체 |
| **= mButtonRefresh** | Button | 게시물 리스트를 갱신하기 위한 객체 |
| **= mButtonPost** | Button | PostActivity 로 인텐트시키는 객체 |
| **= mEditDistance** | EditText | 게시물 리스트를 갱신하는 거리 조건을 담고 있는 객체 |
| **= mMapLocationManager** | NMapLocationManager | 현재 위치를 갱신하기 위해 사용 |
| **= myLocation** | NGeoPoint | 현재 위치 정보를 담고 있는 객체 |

abstract class NMapFrameActivity extends NMapActivity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | frame.xml의 내용을 표현한다. 그리고 loadContent()를 실행한다. |
| **= onResume()** | void | Activity가 Resume 상태가 될 때 호출된다. startMyLocation()을 실행한다. |
| **= onStop()** | void | Activity가 Stop 상태가 될 때 호출된다. stopMyLocation()을 실행한다. |
| **= startMyLocation()** | void | mMapLocationManager의 위치 탐색을 시작한다. 만약 사용이 불가능한 상태라면, GPS 설정 화면으로 넘어간다. |
| **= stopMyLocation()** | void | 위치 탐색을 종료한다. |
| **+ onLocationChanged( NMapLocationManager, NGeoPoint)** | void | myLocatioin을 갱신한다. 그리고 presentLocationChanged()를 실핸한다. |
| **+ onLocationUpdateTimeout( NMapLocationManager)** | void | stopMyLocation()을 실행한다. 그리고 에러 메세지를 출력한다. |
| **= abstract loadContent()** | void | 페이지가 만들어질 때 실행된다. |
| **= abstract presentPosts()** | void | 게시물을 표현한다. |
| **= abstract presentLocationChanged()** | Void | 위치가 변할 때 호출된다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **= mInflater** | LayoutInFlater | Layout xml 파일을 뷰로 가져오기 위한 객체 |
| **= mViewgroup** | ViewGroup | 상속받은 클래스의 뷰를 담기 위한 객체 |
| **= mButtonPosts** | ImageButton | PostsActivity 로 인텐트시키는 객체 |
| **= mButtonMap** | ImageButton | MapActivity 로 인텐트시키는 객체 |
| **= mButtonAR** | ImageButton | ARActivity 로 인텐트시키는 객체 |
| **= mButtonMember** | ImageButton | MemberinfoActivity 로 인텐트시키는 객체 |
| **= mButtonRefresh** | Button | 게시물 리스트를 갱신하기 위한 객체 |
| **= mButtonPost** | Button | PostActivity 로 인텐트시키는 객체 |
| **= mEditDistance** | EditText | 게시물 리스트를 갱신하는 거리 조건을 담고 있는 객체 |
| **= mMapLocationManager** | NMapLocationManager | 현재 위치를 갱신하기 위해 사용 |
| **= myLocation** | NGeoPoint | 현재 위치 정보를 담고 있는 객체 |

class LoginActivity extends Activity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | login.xml의 내용을 표현한다. |
| **= onActivityResult()** | Void | ConfigureDialog의 결과로 실행된다. SharedPreferences에 Server의 Ip, Port를 저장한다. |
| **+ onCreateOptionsMenu( Menu)** | boolean | 메뉴를 생성한다. |
| **+ onOptionsItemSelected( MenuItem)** | void | 메뉴가 선택되면 호출된다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **= mEditId** | EditText | 로그인 시 사용되는 아이디 값을 담는 객체 |
| **= mEditPw** | EditText | 로그인 시 사용되는 암호 값을 담는 객체 |
| **= mButtonLogin** | ImageButton | 로그인을 실행하기 위한 객체 |
| **= mButtonJoin** | ImageButton | MemeberJoinActivity 로 인텐트시키는 객체 |

class MapActivity extends NMapFrameActivity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **= loadContent()** | void | map.xml의 내용을 표현한다. 그리고 startMyLocation(), presentPosts() 을 실행한다. |
| **= startMyLocation()** | void | 현재 위치를 파악하고, 현재 위치를 표현한다. 만약 사용이 불가능한 상태라면, GPS 설정 화면으로 넘어간다. |
| **= stopMyLocation()** | void | 위치 탐색을 종료한다. 그리고 현재 위치 표현을 삭제한다. |
| **= presentLocationChanged()** | void | 현재 위치가 변함을 표현한다. 실제로는 어떠한 일도 하지 않는다. |
| **= presentPosts()** | void | 게시물을 표현한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- (s) (f) API\_KEY** | String |  |
| **- mMapContainerView** | MapContainerView |  |
| **- mMapView** | NMapView |  |
| **- mMapController** | NMapController |  |
| **- (s) (f) NMAP\_LOCATION\_DEFAULT** | NGeoPoint |  |
| **- mOverlayManager** | NMapOverlayManager |  |
| **- mMyLocationOverlay** | NMapMyLocationOverlay |  |
| **- mMapCompassManager** | NMapCompassManager |  |
| **- mMapViewerResourceProvider** | NMapViewerResourceProvider |  |
| **- mPoiDataOverlay** | NMapPOIdataOverlay |  |
| **- mCompassView** | CompassView |  |
| **- onCalloutOverlayListener** | NMapOverlayManager. OnCalloutOverlayListener |  |
| **- onMapViewStateChangeListener** | NMapView. OnMapStateChangeListener |  |
| **- onMapViewTouchDelegate** | NMapView. OnMapViewDelegate |  |
| **- onPOIdataStateChangeListener** | NMapPOIdataOverlay. OnStateChangeListener |  |

class MemberinfoActivity extends Activity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | memberinfo.xml의 내용을 표현한다. 그리고 loadUser()을 실행한다. |
| **+ onClick(View)** | void | 클릭 액션에 따른 이벤트를 정의한다. |
| **= onActivityResult(int, int, Intent)** | void | android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE의 결과로 실행된다. 카메라에서 얻어온 이미지를 출력한다. |
| **- loadUser()** | void | 서버로부터 로그인된 사용자의 정보를 불러와 출력한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mButtonSend** | Button | 입력된 결과를 서버로 전송시키는 객체 |
| **- mButtonTakePhoto** | Button | android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE로 인텐트시키는 객체 |
| **- mEditId** | EditText | 아이디 정보를 담는 객체 |
| **- mEditPw** | EditText | 암호 정보를 담는 객체 |
| **- mEditPw2** | EditText | 암호 확인 정보를 담는 객체 |
| **- mEditName** | EditText | 이름 정보를 담는 객체 |
| **- mEditAge** | EditText | 나이 정보를 담는 객체 |
| **- mEditSex** | EditText | 성별 정보를 담는 객체 |
| **- mEditPhone** | EditText | 전화번호 정보를 담는 객체 |
| **- mEditJob** | EditText | 직업 정보를 담는 객체 |
| **- mImagePhoto** | ImageView | 사진을 표현하는 객체 |

class MemberJoinActivity extends Activity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | memberjoin.xml의 내용을 표현한다. |
| **+ onClick(View)** | void | 클릭 액션에 따른 이벤트를 정의한다. |
| **= onActivityResult(int, int, Intent)** | void | android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE의 결과로 실행된다. 카메라에서 얻어온 이미지를 출력한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mButtonSend** | Button | 입력된 결과를 서버로 전송시키는 객체 |
| **- mButtonTakePhoto** | Button | android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE로 인텐트시키는 객체 |
| **- mEditId** | EditText | 아이디 정보를 담는 객체 |
| **- mEditPw** | EditText | 암호 정보를 담는 객체 |
| **- mEditPw2** | EditText | 암호 확인 정보를 담는 객체 |
| **- mEditName** | EditText | 이름 정보를 담는 객체 |
| **- mEditAge** | EditText | 나이 정보를 담는 객체 |
| **- mEditSex** | EditText | 성별 정보를 담는 객체 |
| **- mEditPhone** | EditText | 전화번호 정보를 담는 객체 |
| **- mEditJob** | EditText | 직업 정보를 담는 객체 |
| **- mImagePhoto** | ImageView | 사진을 표현하는 객체 |

class PostActivity extends Activity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | post.xml의 내용을 표현한다. |
| **+ onClick(View)** | void | 클릭 액션에 따른 이벤트를 정의한다. |
| **= onActivityResult(int, int, Intent)** | void | android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE의 결과로 실행된다. 카메라에서 얻어온 이미지를 출력한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mButtonSend** | Button | 입력된 결과를 서버로 전송시키는 객체 |
| **- mButtonTakePhoto** | Button | android.provider.MediaStore.ACTION\_IMAGE\_CAPTURE로 인텐트시키는 객체 |
| **- mEditId** | EditText | 아이디 정보를 담는 객체 |
| **- mEditComment** | EditText | 코멘트 정보를 담는 객체 |
| **- mImagePhoto** | ImageView | 사진을 표현하는 객체 |
| **- longitude** | double | 현재 위치의 경도 정보를 가진다. |
| **- latitude** | double | 현재 위치의 위도 정보를 가진다. |

class PostsActivity extends FrameActivity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **= loadContent()** | void | posts.xml의 내용을 표현한다. 그리고 presentPosts()를 실행한다. |
| **= presentPosts()** | void | 게시물을 리스트로 표현한다. |
| **= presentLocationiChanged()** | void | 각 게시물과의 거리를 갱신한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mListViewPosts** | ListView | 게시물을 표현하기 위한 리스트 객체 |

|  |
| --- |
| **Nested Class** |
| **class PostAdapter extends BaseAdapter**   |  |  | | --- | --- | | **Constuctor** | **Description** | | **PostAdapter(Context, in, List<PostBean>)** |  |  |  |  |  | | --- | --- | --- | | **Method** | **Return Type** | **Description** | | **+ getCount()** | int | 리스트의 총 개수를 반환한다. | | **+ getItem(int)** | Object | 해당 인덱스에 해당하는 행의 객체를 반환한다. | | **+ getItemId(int)** | long | 해당 인덱스에 해당하는 행의 아이디를 반환한다. | | **+ getView(int, View, ViewGroup)** | View | 해당 인덱스에 해당하는 행의 뷰를 반환한다. |  |  |  |  | | --- | --- | --- | | **Field** | **Type** | **Description** | | **context** | Context | ListView가 표현되는 Context | | **id** | int | ListView의 id | | **posts** | List<PostBean> | ListView의 표현 객체 | |

class ConfigureDialog extends Activity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | configure.xml의 내용을 표현한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mEditIp** | EditText | 서버의 Ip 정보를 담는 객체 |
| **- mEditPort** | EditText | 서버의 Port 정보를 담는 객체 |
| **- mButtonSave** | Button | 서버 정보를 저장하는 객체 |

class PostInfoDialog extends Activity

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ onCreate(Bundle)** | void | postinfo.xml의 내용을 표현한다. |
| **+ onClick(View)** | voic | 클릭 액션에 따른 이벤트를 정의한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mTextId** | TextView | 게시물의 ID 정보를 담는 객체 |
| **- mTextComment** | TextView | 게시물의 Comment 정보를 담는 객체 |
| **- mImage** | ImageView | 게시물의 Image 정보를 담는 객체 |
| **- mButtonCall** | Button | 게시물의 사용자에게 전화를 거는 Intent를 실행하는  객체 |
| **- mButtonLetter** | Button | 게시물의 사용자에게 문자를 보내는 Intent를 실행하는 객체 |

class CameraView extends SurfaceView

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **+ CameraView(Context)** |  |
| **+ CameraView(Context, AttributeSet)** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ surfaceCreated(SurfaceHolder)** | Void | Camera 객체의 Preview를 시작한다. |
| **+ surfaceChanged( SurfaceHolder, int, int, int)** | void | 변경된 크기로 Camera 객체의 Preview를 시작한다. |
| **+ surfaceDestroyed(SurfaceHolder)** | void | Camera 객체의 Preview를 종료한 후, 자원을 반환한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mCamera** | Camera | Camera 를 이용하기 위한 객체 |
| **- mSurfaceHolder** | SurfaceHolder | SurfaceView의 상태를 캐치하기 위한 객체 |

class CompassView extends ImageButton

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **+ CompassView(Context)** |  |
| **+ CompassView(Context, AttributeSet)** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ setCompass(Compass)** | void | Compass에 Listener를 등록한다. |
| **+ setBackgroundBitmap(Bitmap)** | void | 나침반의 배경이 될 이미지를 링크한다. |
| **+ setOrientationBitmap(Bitmap)** | void | 나침반이 가르키는 방향을 나타낼 이미지를 링크한다. |
| **= onDraw(Canvas)** | void | 나침반을 그린다. |
| **+ onSensorChanged(SensorEvent)** | void | mValues 를 수정한다. |
| **+ onAccuracyChanged(Sensor, int)** | void |  |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mBitmapBackground** | Bitmap | 나침반의 배경이 되는 이미지 객체 |
| **- mBitmapOrient** | Bitmap | 나침반의 바늘이 되는 이미지 객체 |
| **- mValues** | double[] | 나침반의 각도 정보를 가지고 있는 배열 |

Sensor Classes

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| **Compass** | Orientation 센서를 등록하고 SensorEventListener 를 관리하는 클래스 |

class Compass

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **+ Compass(Context)** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ addListener( SensorEventListener)** | void | mListSensorListeners 에 SensorEventListener 를 추가한다. |
| **+ removeListener( SensorEventListener)** | void | mListSensorListeners 에 SensorEventListener 를 삭제한다. |
| **+ start()** | Void | 모든 SensorEventListener 를 등록한다. |
| **+ stop()** | void | 모든 SensorEventListener 를 제거한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- mSensorManager** | SensorManager | 장비의 센서에 접근하게 해주는 객체 |
| **- mSensor** | Sensor | Orientation 센서를 나타내는 객체 |
| **- mListSensorListeners** | List<SensorEventListener> | Sensor 의 값이 바뀔 때, SensorManager로부터 알려주는 리스너의 집합 |

Utillity Classes

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| **Calculations** | 좌표 계산을 위한 클래스 |
| **GraphicUtils** | Image File, Bitmap, Drawable 변환을 위한 클래스 |

Class Calculations

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **- Calculations ()** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ (s) distanceBetween3DPoint( double[], double[])** | double | 두 좌표 사이의 거리를 반환한다. |
| **+ (s) distanceBetweenTwoGeoPoint( NGeoPoint, NGeoPoint)** | int | 두 좌표 사이의 실제 거리를 반환한다. |
| **+ (s) distanceFromZero( double[])** | double | 원점에서 해당 좌표까지의 거리를 반환한다. |
| **+ (s) getLocationOfViewOnScreen( float[], NGeoPoint, NGeoPoint)** | double[] | 보는 방향을 기준으로 목표체의 새로운 좌표를 반환한다. |
| **+ (s) xRotationConvert(double[], double)** | double[] | x축으로 회전한 후의 변환된 좌표를 반환한다. |
| **+ (s) yRotationConvert(double[], double)** | double[] | y축으로 회전한 후의 변환된 좌표를 반환한다. |
| **+ (s) zRotationConvert(double[], double)** | double[] | z축으로 회전한 후의 변환된 좌표를 반환한다. |

Class GraphlicUtils

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **- GraphlicUtils()** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ createBitmapFromImage(File)** | Bitmap | 파일 객체로부터 Bitmap 객체를 만들어 반환한다. |
| **+ createImageFile(Context, byte[])** | File | 배열을 Context에서 지정한 캐시 폴더에 파일로 저장한 후 반환한다. |
| **+ createTempImageFile(Context, Drawable)** | File | Drawable 객체을 Context에서 지정한 캐시 폴더에 파일로 저장한 후 반환한다. |
| **+ getOverlayDrawableFromBitmap( Bitmap)** | Drawable | Bitmap 객체를 Drawable 객체로 변환 후, Overlay 형식에 맞게 이미지를 조정 후, 반환한다. |
| **+ setDefaultNoSearchImage(Resources, int)** | Void | getOverlayDrawableFromBitmap(Bitmap) 메서드의 반환 중, Bitmap이 Null 이 경우 반환되는 Drawable 의 형태를 결정한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **+ (s) (f) TEMP\_IMAGE\_FILE\_NAME** | String | 임시 파일을 생성할 때 사용되는 파일 이름 : “temp.jpg” |
| **+ (s) DEFAULT\_NO\_SEARCH\_IMAGE** | Drawable | getOverlayDrawableFromBitmap(Bitmap) 메서드의 반환 중, Bitmap이 Null 이 경우 반환되는 객체 |
| **- (s) (f) POIOVERLAY\_IMAGE\_WIDTH** | int | Overlay Drawable을 생성할 때, 사용되는 Overlay의 넓이 : 150 |
| **- (s) (f) POIOVERLAY\_IMAGE\_HEIGHT** | int | Overlay Drawable을 생성할 때, 사용되는 Overlay의 높이 : 150 |

# Service Server

### 디렉토리 구성

|  |  |
| --- | --- |
| **Directory Name** | **Description** |
| **src** | \*.java Files |
| **lib** | External Library |
| **bin** | \*.class Files |
| **properties** | Properties Text Files |
| **Image** | Image Files(서버 구동 시, 클라이언트에서 받아오는 이미지 파일들) |

### 패키지 구성

|  |  |
| --- | --- |
| **Packege Name** | **Description** |
| **kr.android.hellodrinking.server** | 서버의 생성 및 기능에 관련한 클래스를 제공합니다. |
| **kr.android.hellodrinking.server.db** | 데이터 베이스의 연결 및 접근에 관련한 클래스를 제공합니다. |
| **kr.android.hellodrinking.server.db.connection** | 데이터 베이스의 연결에 관련한 클래스를 제공합니다. |
| **kr.android.hellodrinking.transmission** | 네트워크 어플리케이션의 구현을 위한 클래스를 제공합니다. |
| **kr.android.hellodrinking.transmission.dto** | Bean 의 생성에 필요한 클래스를 제공합니다. |
| **kr.android.hellodrinking.transmission.exception** | 통신 중 일어날 수 있는 예외에 대해 정의한 클래스를 제공합니다. |

클래스 관계 개요도

NMapFrameActivity

ARActivity

PostsActivity

MapActivity

PostActivity

LoginActivity

MemberJoinActivity

MemberInfoActivity

|  |  |  |
| --- | --- | --- |
| **Class** | **Description** | **해당 Class** |
| **Server** | Serversocket을 맴버로 가지고 있으며, 클라이언트 접속이 있을 시, ClientThread를 생성 | HelloDrinkingServer |
| **ClientThread** | 접속한 Client의 요청에 따른 처리를 담당하는 클래스 | ClientThread, RequestProcess |
| **DAO** | 데이터베이스에 접속하여 해당 처리를 수행하는 클래스 | DBAccess, Connectable, ConnectFromSettingFile |
| **Data Model** | Activity 또는 View 에 표현되는 정보를 담고 있는 클래스  \* 클라이언트와 동일한 클래스가 사용된다. | Bean, PostBean, UserBean |
| **Request** | 메서드를 호출하기 위해서 필요한 정보를 포함한 클래스  \* 클라이언트와 동일한 클래스가 사용된다. | Request, Requestable, RequestBeanPackege |
| **Responce** | Request를 받아 처리된 정보를 포함한 클래스  \* 클라이언트와 동일한 클래스가 사용된다. | ResponceBeanPackege |

Server Classes

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| **HelloDrinkingServer** | Serversocket을 맴버로 가지고 있으며, 클라이언트 접속이 있을 시, ClientThread를 생성하는 클래스 |

class HelloDrinkingServer

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **HelloDrinkingServer()** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ start()** | void | 클라이언트 접속이 있을 시, ClientThread를 생성한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **+ (s) (f) DEFAULT\_IMAGEFILE\_DIRECTORY** | String | 이미지 파일이 저장되는 디렉토리 패스 : Image |
| **+ (s) (f) PROFERTIES\_FILE\_PATH** | String | 설정 파일이 저장되는 디렉토리 패스 : properties/properties.txt |
| **+ (s) (f) DEFAULT\_DB\_PORT** | int | 기본 DB port : 1521 |
| **+ (s) (f) DEFAULT\_DB\_IP** | String | 기본 DB ip : 127.0.0.1 |
| **+ (s) (f) DEFAULT\_DB\_USER** | String | 기본 DB user : Project |
| **+ (s) (f) DEFAULT\_DB\_PASSWOR** | String | 기본 DB password :kitri |
| **+ (s) (f) DEFAULT\_JDBC\_ID** | String | 기본 JDBC id : xe |
| **+ (s) (f) DEFAULT\_JDBC\_CLASS** | String | 기본 JDBC class : oracle.jdbc.driver.OracleDriver |
| **+ (s) (f) DEFAULT\_JDBC\_DRIVER** | String | 기본 JDBC driver : oracle:thin |

ClientThread Classes

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| **ClientThread** | Socket 연결 및 IO 관리에 사용하는 클래스 |
| **RequestProcess** | Client의 Request 처리에 사용되는 클래스 |

class ClientThread extends Thread

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **ClientThread(Socket)** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ run()** | void | 요청에 따라 RequestProcess를 생성 및 실행한다. |
| **+ close()** | void | 모든 자원을 반환한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **mSocket** | Socket | 클라이언트 소켓를 생성 및 관리에 사용되는 객체 |
| **mReader** | InputStream | 소켓의 스트림 관리에 사용되는 객체 |
| **mWriter** | OutputStream | 소켓의 스트림 관리에 사용되는 객체 |
| **mObjectReader** | ObjectInputStream | 소켓의 스트림 관리에 사용되는 객체 |
| **mObjectWriter** | ObjectOutputStream | 소켓의 스트림 관리에 사용되는 객체 |
| **mConnection** | Connection | DB 연결에 사용되는 객체 |

class RequestProcess

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **RequestProcess(RequestBeanPackege, Connection, ObjectOutputStream)** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ processing()** | void | Request의 종류에 따른 분기를 실행한다. |
| **- register()** | boolean | 회원 가입을 위한 메서드 |
| **- login()** | boolean | 로그인을 위한 메서드 |
| **- userModify()** | boolean | 회원 정보 수정을 위한 메서드 |
| **- getUser()** | boolean | 회원의 정보를 받아오기 위한 메서드 |
| **- post()** | boolean | 게시물을 등록하기 위한 메서드 |
| **- getPosts()** | boolean | 게시물을 가져오기 위한 메서드 |
| **- writerImageFileAndGetImageFilePath( String, byte[])** | String | 파일을 기록하고, 해당 파일의 경로를 반환한다. |
| **sendResponce(ResponceBeanPackege)** | boolean | ResponcePackege를 클라이언트에 전송한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **mRequest** | Request | 요청 처리를 위해 사용되는 객체 |
| **mObjectWriter** | ObjectOutputStream | 소켓의 스트림 쓰기에 사용되는 객체 |
| **mDBAccess** | DBAccess | 데이터베이스에 접근하여 해당 요청을 처리하기 위해 사용되는 객체 |

DAO Classes

ConfigureDialog

UserInfoDialog

CameraView

|  |  |
| --- | --- |
| **Class Name** | **Description** |
| **DBAccess** | 데이터베이스에 접근하는 DAO 클래스 |
| **ConnectionFromSettingFile** | 설정 파일로부터 정보를 읽어 Connection 객체를 생성하는 클래스 |
| **Connectable** | Connection 객체를 생성하기 위해 필요한 메서드를 정의한 인터페이스 |

interface Connectable

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ getConnection()** | Connection | DB연결에 사용되는 Connection 객체를 반환한다. |

class ConnectionFromSettingFile

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **- ConnectionFromSettingFile()** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ (s) getInstance()** | ConnectFrom SettingFile | ConnectFromSettingFile 의 생성을 대신하여 사용된다. |
| **+ getConnection()** | Connection | DB연결에 사용되는 Connection 객체를 반환한다. |
| **- createDefaultFile(File)** | void | HelloDrinkingServer의 변수를 통해 설정 파일을 만든다. |
| **- setDefaultSetting()** | void | HelloDrinkingServer의 변수를 통해 Connection 객체 생성에 필요한 변수를 초기화한다. |
| **- readSettingFile(File)** | void | 해당 파일 객체로부터 설정에 Connection 객체 생성에 필요한 변수를 초기화한다. |
| **- getValue(StreamTokenizer)** | String | StreamTokenizer 객체로부터 String 값을 구해 반환한다. |
| **- readDBSetting(BufferedReader)** | String | DB 설정에 관련한 변수를 초기화한다.  다음 행을 반환한다. |
| **- readJDBCSetting(BufferedReader)** | String | JDBC 설정에 관련한 변수를 초기화한다.  다음 행을 반환한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- (s) instance** | ConnectFromSettingFile |  |
| **- db\_ip** | String | Connection 객체의 생성에 사용되는 DB ip |
| **- db\_port** | int | Connection 객체의 생성에 사용되는 DB port |
| **- db\_user** | String | Connection 객체의 생성에 사용되는 DB user |
| **- db\_password** | String | Connection 객체의 생성에 사용되는 DB password |
| **- jdbc\_class** | String | Connection 객체의 생성에 사용되는 JDBC class |
| **- jdbc\_driver** | String | Connection 객체의 생성에 사용되는 JDBC driver |
| **- jdbc\_id** | String | Connection 객체의 생성에 사용되는 JDBC id |

class DBAccess

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **DBAccess (Connection)** |  |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ addLoginHistory(String, boolean)** | Exception | 로그인 기록을 추가한다. 시간, 아이디, 성공여부를 기록한다. |
| **+ getPosts(double, double, int)** | Exceptioni | 지정된 경도, 위도에서 일정 거리 내의 게시물을 가져온다.  성공적으로 가져온 경우 kr.android. hellodrinking.transmission.exception.Message를 반환한다. |
| **+ getUser(String)** | Exceptioni | 아이디를 통해 검색된 User의 정보를 가져온다.  성공적으로 가져온 경우 kr.android. hellodrinking.transmission.exception.Message를 반환한다. |
| **+ getUserPassword(String)** | Exceptioni | 아이디를 통해 검색된 User의 암호를 가져온다.  성공적으로 가져온 경우 kr.android. hellodrinking.transmission.exception.Message를 반환한다. |
| **+ register(String, String, String, String, String, String, String, String)** | Exceptioni | User를 등록한다. |
| **+ post(String, String, String, double, double)** | Exceptioni | 게시물을 등록한다. |
| **+ setImageAtPostInfo(int, String)** | Exceptioni | 지정 postnum의 게시물의 Image를 해당 파일 패스로 교체한다. |
| **+ setImageAtUserInfo(String, String)** | Exceptioni | 지정 id의 사용자의 Image를 해당 파일 패스로 교체한다. |
| **+ userModify(String, String, String, String, String, String, String, String)** | Exceptioni | 지정 id의 사용자의 정보를 갱신한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **mConnection** | Connection | DB 연결에 사용되는 객체 |

Request Classes

CompassView

MapContainerView

Server

|  |  |
| --- | --- |
| **Class Name** | **Descripstion** |
| **Requestable** | 가능한 요청의 종류를 정의하는 인터페이스 |
| **Request** | 요청을 구현하는 클래스 |
| **RequestBeanPackege** | 실제 요청되는 내부 메시지(Bean) 클래스 |

interface Requestable

* 모든 요청은 OptionalDataException, IOException 을 발생시킨다.

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ getPosts(double, double, int)** | ResoponceBeanPackege | 지정된 경도, 위도로부터 지정 거리 내의 게시물을 가져온다. |
| **+ getUser(String)** | ResoponceBeanPackege | 지정된 id를 가진 사용자의 정보를 가져온다. |
| **+ getUser(UserBean)** | ResoponceBeanPackege | 지정된 id를 가진 사용자의 정보를 가져온다. |
| **+ login(String, String)** | ResoponceBeanPackege | 지정된 id와 password가 옳바른지 확인한다. |
| **+ login(UserBean)** | ResoponceBeanPackege | 지정된 id와 password가 옳바른지 확인한다. |
| **+ modifyUser(String, String, String, String, String, String, String, String)** | ResoponceBeanPackege | 지정 id의 사용자의 정보를 갱신한다. |
| **+ modifyUser(UserBean)** | ResoponceBeanPackege | 지정 id의 사용자의 정보를 갱신한다. |
| **+ post(PostBean)** | ResoponceBeanPackege | 게시물을 등록한다. |
| **+ post(String, String, String, double, double)** | ResoponceBeanPackege | 게시물을 등록한다. |
| **+ register(String, String, String, String, String, String, String, String)** | ResoponceBeanPackege | 사용자를 등록한다. |
| **+ register(UserBean)** | ResoponceBeanPackege | 사용자를 등록한다. |

class Request

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| **Request()** | 기본 ip, port의 서버에 접근한다. |
| **Request(String, int)** | 지정 ip, port의 서버에 접근한다. |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ getPosts(double, double, int)** | ResoponceBeanPackege | 지정된 경도, 위도로부터 지정 거리 내의 게시물을 가져온다. |
| **+ getUser(String)** | ResoponceBeanPackege | 지정된 id를 가진 사용자의 정보를 가져온다. |
| **+ getUser(UserBean)** | ResoponceBeanPackege | 지정된 id를 가진 사용자의 정보를 가져온다. |
| **+ login(String, String)** | ResoponceBeanPackege | 지정된 id와 password가 옳바른지 확인한다. |
| **+ login(UserBean)** | ResoponceBeanPackege | 지정된 id와 password가 옳바른지 확인한다. |
| **+ modifyUser(String, String, String, String, String, String, String, String)** | ResoponceBeanPackege | 지정 id의 사용자의 정보를 갱신한다. |
| **+ modifyUser(UserBean)** | ResoponceBeanPackege | 지정 id의 사용자의 정보를 갱신한다. |
| **+ post(PostBean)** | ResoponceBeanPackege | 게시물을 등록한다. |
| **+ post(String, String, String, double, double)** | ResoponceBeanPackege | 게시물을 등록한다. |
| **+ register(String, String, String, String, String, String, String, String)** | ResoponceBeanPackege | 사용자를 등록한다. |
| **+ register(UserBean)** | ResoponceBeanPackege | 사용자를 등록한다. |
| **- createSocket()** | void | Socket 객체를 생성한다. |
| **- sendRequestAndGetResponce( RequestBeanPackege)** | ResoponceBeanPackege | 지정된 RequestBeanPackege를 전송 후, ResponceBeanPackege를 전송 받아 반환한다. |
| **+ close()** | void | 모든 자원을 반환한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **mSocket** | Socket | 클라이언트 소켓를 생성 및 관리에 사용되는 객체 |
| **mReader** | InputStream | 소켓의 스트림 관리에 사용되는 객체 |
| **mWriter** | OutputStream | 소켓의 스트림 관리에 사용되는 객체 |
| **mObjectReader** | ObjectInputStream | 소켓의 스트림 관리에 사용되는 객체 |
| **mObjectWriter** | ObjectOutputStream | 소켓의 스트림 관리에 사용되는 객체 |

Data Model Classes

ClientThread

DAO

Resonce

|  |  |
| --- | --- |
| **Class Name** | **Descripstion** |
| **Bean** | UserBean과 PostBean에서 공통적으로 사용되는 필드 및 메서드를 추상화시킨 추상 클래스 |
| **UserBean** | 사용자에 관련한 정보를 담고 있는 클래스 |
| **PostBean** | 게시물에 관련한 정보를 담고 있는 클래스 |

abstract class Bean implements Serializable

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ getBuffer()** | byte[] | 파일 배열을 반환한다. |
| **+ getId()** | String | 아이디를 반환한다. |
| **+ getImageFilePath()** | String | 이미지 파일 경로를 반환한다. |
| **+ setBuffer(byte[])** | void | 파일 배열을 설정한다. |
| **+ setId(String)** | void | 아이디를 설정한다. |
| **+ setImageFilePath(String)** | void | 이미지 파일 경로를 설정한다. 그리고 convertToBytes()를 실행한다. |
| **= convertToBytes()** | void | 파일 경로를 통해 파일을 읽어 그 배열을 buffer에 저장시킨다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **= buffer** | byte[] | 파일의 내용을 담고 있는 배열 |
| **= id** | String | 아이디를 가르키는 객체 |
| **= imageFilePath** | String | 파일 경로를 가르키는 객체 |

class UserBean extends Bean implements Serializable

|  |  |
| --- | --- |
| **Constructor** | **Descripstion** |
| **+ UserBean()** |  |
| **+ UserBean(String)** | 아이디를 가지고 객체를 생성한다. |
| **+ UserBean(String, String)** | 아이디와 암호를 가지고 객체를 생성한다. |
| **+ UserBean(String, String, String, String, String, String, String, String)** | 모든 정보를 가지고 객체를 생성한다. |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ getAge()** | String | 나이를 반환한다. |
| **+ getJob()** | String | 직업을 반환한다. |
| **+ getName()** | String | 이름을 반환한다. |
| **+ getPassword()** | String | 암호를 반환한다. |
| **+ getPhone()** | String | 전화번호를 반환한다. |
| **+ getSex()** | String | 성별을 반환한다. |
| **+ setAge(String)** | void | 나이를 설정한다. |
| **+ setJob(String)** | void | 직업를 설정한다. |
| **+ setName(String)** | void | 이름를 설정한다. |
| **+ setPassword(String)** | void | 암호를 설정한다. |
| **+ setPhone(String)** | void | 전환번호를 설정한다. |
| **+ setSex(String)** | void | 성별를 설정한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- age** | String | 나이를 가르키는 객체 |
| **- job** | String | 직업을 가르키는 객체 |
| **- name** | String | 이름을 가르키는 객체 |
| **- password** | String | 암호를 가르키는 객체 |
| **- phone** | String | 전화번호를 가르키는 객체 |
| **- sex** | String | 성별을 가르키는 객체 |

class PostBean extends Bean implements Serializable

|  |  |
| --- | --- |
| **Constructor** | **Descripstion** |
| **+ PostBean(String, double, double)** | 아이디, 경도, 위도 정보를 가지고 객체를 생성한다. |
| **+ PostBean(String, String, String, double, double)** | 모든 정보를 가지고 객체를 생성한다. |

|  |  |  |
| --- | --- | --- |
| **Method** | **Return Type** | **Description** |
| **+ getComment()** | String | 코멘트를 반환한다. |
| **+ getLatitude()** | double | 위도을 반환한다. |
| **+ getLongitude()** | double | 경도을 반환한다. |
| **+ getPostnum()** | int | 게시물번호를 반환한다. |
| **+ setComment(String)** | void | 코멘트를 설정한다. |
| **+ setLatitude(double)** | void | 위도를 설정한다. |
| **+ setLongitude(double)** | void | 경도를 설정한다. |
| **+ setPostnum(int)** | void | 게시물번호를 설정한다. |

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| **- comment** | String | 나이를 가르키는 객체 |
| **- latitude** | double | 위도을 가르키는 객체 |
| **- longitude** | double | 경도을 가르키는 객체 |
| **- postnum** | int | 게시물번호를 가르키는 객체 |

# 결론

# 미 해결 문제점

* Network 자원을 활용한 채팅 구현
* 서버의 커넥션 풀 문제
* 파일 전송 오버헤드 문제
* 중복적인 코드
* 세부적 UI 구현

# 참고자료

# 네이버 지도 API : http://dev.naver.com/openapi/apis/map/android/example

# 센서 카메라 위치정보 활용 - 박유태

# Mixare : http://code.google.com/p/mixare/

# Wikitude API 1.1 : http://www.wikitude.org

# hi there : 안드로이드 마켓

# 이미지 참고 자료

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 그림 | 사용위치 | 타입 | 만든일자 | 참조 |
| 234652674v6_480x480_Front_Color-Black |  |  |  | <http://www.cafepress.com/>+beer\_drinking\_penguin\_keepsake\_box,234652674 |
| 아이콘 | 아이콘 | 아이콘 |  |  |
| 뻐꾸기 | 프로젝트 최종보고서 | 이미지 | 2011년 8월30일 | http://rityn.tistory.com/169 |
| 두근두근 우체통 | 프로젝트 최종보고서 | 이미지 | 2011년 8월30일 | <http://rityn.tistory.com/169> |
| 살랑 살랑 돛단배 | 프로젝트 최종보고서 | 이미지 | 2011년 8월30일 | http://ilime.tistory.com/300 |
| 살랑 살랑 돛단배2 | 프로젝트 최종보고서 | 이미지 | 2011년 8월30일 | <http://ilime.tistory.com/300> |
| 부킹호프 | 프로젝트 최종보고서 | 이미지 | 2011년 8월30일 | <http://www.segye.com/Articles/News/Economy/Article.asp?aid=20110607000332&ctg1=01&ctg2=&subctg1=01&subctg2=&cid=0101030100000> |
| 스마트폰의 LBS | 프로젝트 최종보고서 | 이미지 | 2011년 8월30일 | <http://www.bing.com/images/search?q=%ec%8a%a4%eb%a7%88%ed%8a%b8%ed%8f%b0%ea%b3%bc+%ec%84%9c%eb%b2%84+%ed%86%b5%ec%8b%a0&view=detail&id=30AE589DCADC095DCFCF579F8B744E36FD872133&first=0&FORM=IDFRIR> |
| db 모델링 | 프로젝트 최종보고서 | 이미지 | 2011년 8월31일 | <http://databaser.net/moniwiki/wiki.php/%EA%B0%9C%EB%B0%9C%EB%B0%A9%EB%B2%95%EB%A1%A0%EA%B3%BC%EC%A0%95%EB%B3%B4%EA%B3%B5%ED%95%99> |
| 개발방법론 | 프로젝트 최종보고서 | 이미지 | 2011년 8월31일 | <http://databaser.net/moniwiki/wiki.php/%EA%B0%9C%EB%B0%9C%EB%B0%A9%EB%B2%95%EB%A1%A0%EA%B3%BC%EC%A0%95%EB%B3%B4%EA%B3%B5%ED%95%99> |