bizMOB4(Vue) 개발자가이드
For Mobile

Sam Kim

mobile c&c

Innovative Mobile & Fintech Solutions, Ready to Provide

bizMOB4(Vue)

I. Mobile Platform

✔ Platform 의 필요성

플랫폼을 사용하여 App을 제작하는 것이 다양한 측면에서 월등한 이점을 가지고 있습니다.

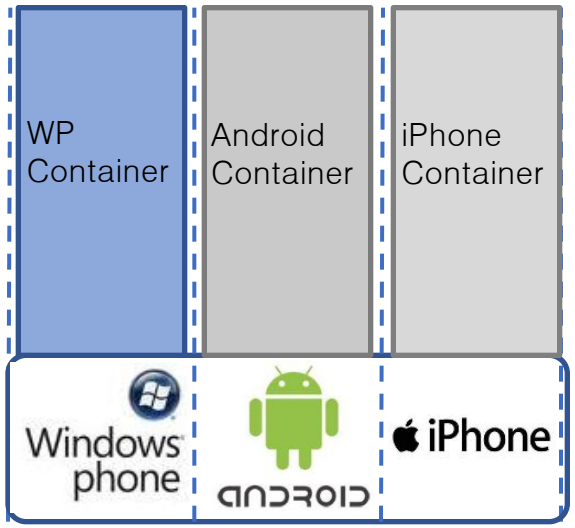
	With Platform	Without Platform
개발 기간	개발 툴 및 기반 SW 제공으로 짧게 걸림	無에서 새로 개발해야 하므로 길게 걸림
품질 보증	이미 검증된 플랫폼 사용하므로 보증가능	개발 결과물의 품질 보증할 수 없음
운영 및 관리 환경	플랫폼에서 운영 및 관리 환경 제공	별도로 개발해야 함
서비스 확장	플랫폼 기반으로 손쉽게 확장 가능	기존 시스템 수정해야 하므로 어려움
유지 보수	플랫폼 제공 업체에서 정기적 패치 적용 개발된 업무 컨텐츠만 유지 보수하면 됨	전체 시스템 소스를 잘 알고 있는 개발자만 유지보수 가능
개발 비용	플랫폼 라이선스가 따로 필요하나, 개발비 감소로 전체 비용은 작음	라이선스는 따로 없으나, 개발비용이 커서 전체 비용은 큼
개발자 수준	주요 기술은 플랫폼에서 제공하므로, 필요한 개발자 수준이 아주 높지 않음	모든 것을 새로 구현해야 하므로 필요한 개발자 수준이 아주 높아야 함
위험 관리	플랫폼에 위험 관리 경험과 기술이 축적되어 예상치 못한 상황에 대한 대처가 쉬움	예상치 못한 상황 발생시 위험 부담 큼
다양한 단말 대응	다양한 OS와 해상도의 단말에 대한 방안이 플랫폼에 녹아 있어 쉽게 대응 가능	다양한 단말에 일일이 대응하여 개발 해야 함
보안	플랫폼에서 보안 정책 제공	보안을 위한 정책 및 SW 별도 개발해야 함

✔ Mobile 플랫폼 Client – 개발기술비교

개발방법	Native App	Mobile Web	Hybrid App, Web
개요	• OS에서 제공하는 기능/UI개발가능 • 성능이 최적화 가능 • 디바이스 기능 사용 유리 • 숙련된 개발자 적고 러닝커브가 길다. 유지 보수 시 상이한 개발 기술자 (안드로이드/아이폰/윈도우 모바일)	• WEB Site 개발 방식과 동일 • 웹 개발자 수급 용이 • 개발 리소스 절약 (One source multiplatform) • OS에 맞는UI개발 불가능 • 보안취약 • 웹 기능외 디바이스 기능 일부 사용 • 배포 시 서비스 중단	• UI는 웹 기술을 이용 • 디바이스 기능은 자바스크립트 API를 이용하여 네이티브 전체기능 사용가능 • 자바스크립트 API는 하이브리드 프레임워크에 의해 제공 • 모바일 웹의 장점인 개발 리소스는 절약됨
UI	Native UI Component (OS Vender provided)	Web (HTML, CSS, JS)	Native UI Component(OS Vender Provided) + Web Component (HTML, CSS, JS)
Communication Protocol	Socket 기반 (HTTP, SOAP, etc)	HTTP	Javascript API + Socket 기반 (HTTP, SOAP, etc)
Data type	text or binary preferred, xml, json	HTML , JSON	HTML, JSON

✔ Mobile 플랫폼 Client – 개발 방식 비교

Native 방식 개발

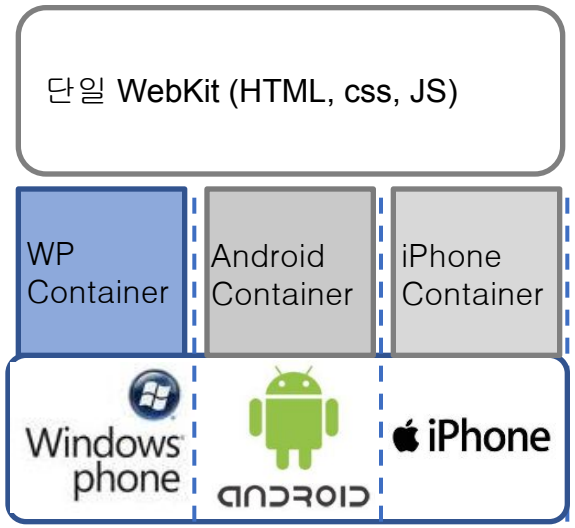


Concept	각 OS에 맞게 전용 App 개발
장점	최적화된 성능, 보안
단점	개발 및 유지보수 고비용 숙련된 개발자 수급 힘듬

Web App 방식 개발

Concept	HTML표준으로 Web 구현 Any Browser 상에서 구동
장점	OS 상관없이 Web 표준으로 PC/모바일 전체 사용가능
단점	낮은 성능, 보안, 단말기능 사용제한

Hybrid 방식 개발



Concept	OS 특화된 UI Container 단일 Web App(HTML)
장점	App 개발시 Web의 장점(UI)과 Native의 장점(기능) 사용 개발인력 多
단점	중간 성능

✔ Mobile 플랫폼 Client – 특장점 비교

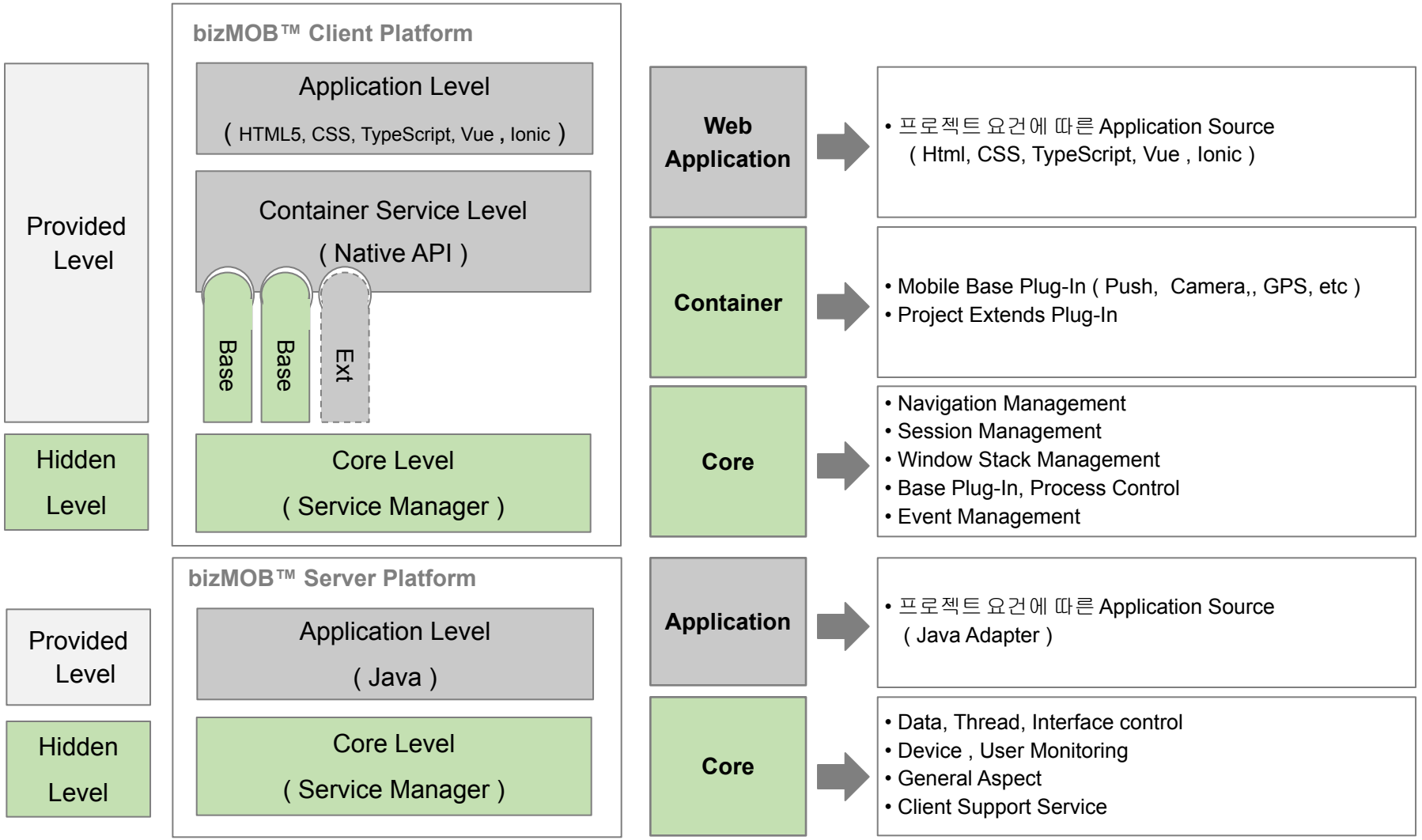
구분	Native App 방식	Web Browsing 방식	Hybrid App 방식
어플리케이션타입	바이너리(OS 별)	웹 어플리케이션 배포파일	App+ Web
배포방법	어플리케이션 배포	필요 없음	최초App은 배포
업그레이드	어플리케이션 업그레이드	웹 서버에 어플리케이션 배포	어플리케이션 업그레이드 + Web Application Update
UI 수준(난이도)	상	하	하
Graphic Performance	상	하	중
어플리케이션개발자	App 개발자(OS별) (Window : C, Android : Java, iOS : Obj.C)	웹 개발자	App 개발자+ 웹 개발자
개발 편의성		웹 지식필요	플랫폼에 대한 지식필요 App + Web 지식
원격삭제	수용가능	불가능	수용가능
복사제어	가능	어려움	가능
파일다운로드제어	가능	어려움	가능
단말원격제어	가능	어려움	가능
Push 서비스	가능	어려움	가능
로컬자원활용	일부가능	어려움	일부 가능
멀티 플랫폼 지원	어려움	쉬움.	중간
Offline Mode	기본지원	불가능	가능
다중 사용자 공동작업	어려움	쉬움	쉬움
키보드Customizing	가능	불가능	가능
단점	OS별 Application 개발	Device Control 제한 Security, Privacy Manage 불가	OS 별 최소의App 개발,재활용성 높음

✔ Framework VS Platform – Cordova 와 플랫폼 비교

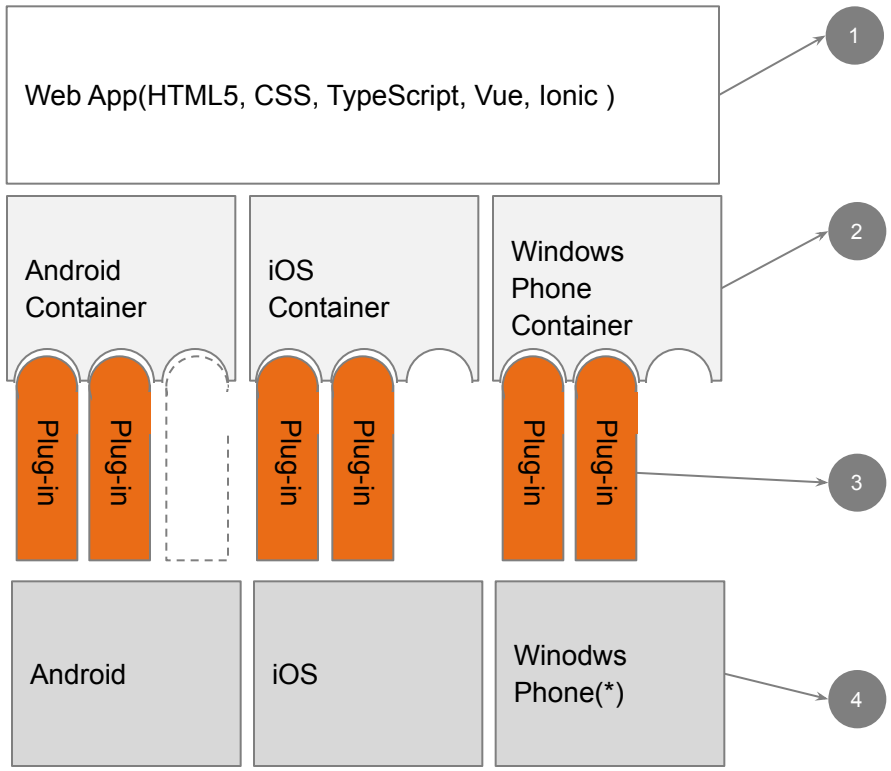
구분	Cordova	Platform
개요	App Framwork	App 개발 및 관리/배포/운영/통계 기능 포함
보안	버전 별 취약점 존재 (일부 버전 오픈마켓 제약)	Cordova 기반 제품 : 버전별 취약점 존재 비 Cordova : 개발사 기준
보안 패치	Cordova 공식 Update 기준	Cordova 기반 제품 : Cordova 공식 Update 기준 비 Cordova : 즉시 지원
배포형태	App내 웹 컨텐츠 포함. Native방식과 동일한 App Update통한 배포	개발사 기준
개발도구	각 OS 별 개발 도구(Visual Studio, Eclipse 등) 기반에Cordova SDK 설치	Eclipse , IntelliJ등 일반 개발 도구의 플러그인 설치 (개발사 기준)
개발실행환경	각 OS별 개발 도구에 따른 Runtime	각 제조사별 Emulator
기능수정	PlugIn 형태의 추가 개발	전체 소스 수정 가능
서버모듈	없음	Client 통신 Protocol과 상응하는 Server군 필요

bizMOB Platform

✔ bizMOB Platform – 계층 모델



✔ bizMOB Client Architecture



1. Once-source Multi-use
HTML, CSS, JS로 업무 UI를 개발하여 각 OS별 Container 상에서 동일에서 동작.

2. 단말 및 OS에 특화된 Container
OS 별 : Android, iOS
화면 사이즈별 : 스마트폰, 패드

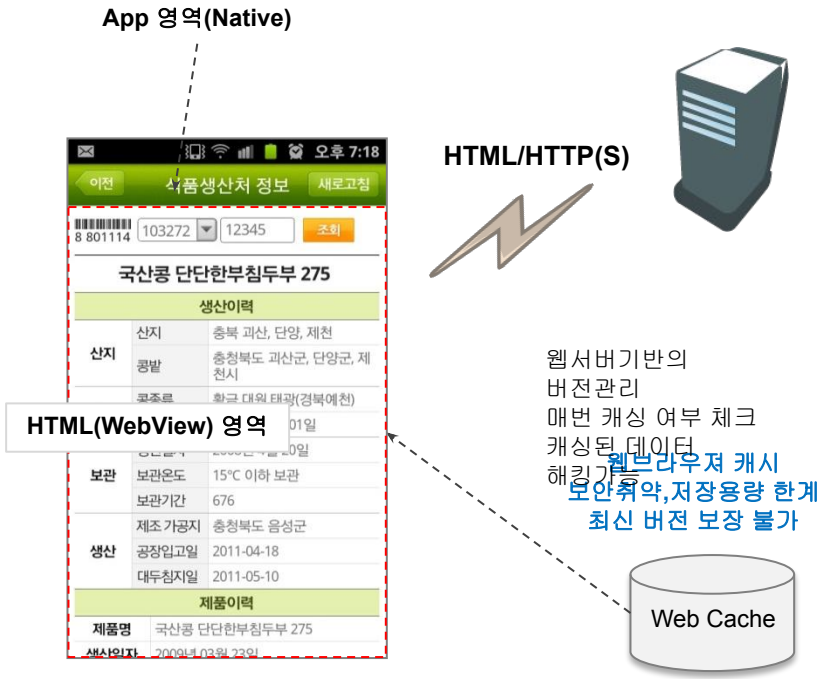
3. Plug-in을 통한 기능 확장
Base Plug-in : 서버통신, 전화, 카메라, 파일시스템, 전화번호부 등
Custom Plug-in : 외부기기 연동, 외부 라이브러리 연동 등

4. OS 버전 업그레이드에 대한 기술 대응
새로운 OS나 버전이 나올 때, 시장의 상황에 맞게 Container 업그레이드 제공

✔ bizMOB Mobile Platform – Hybrid 구성 방식

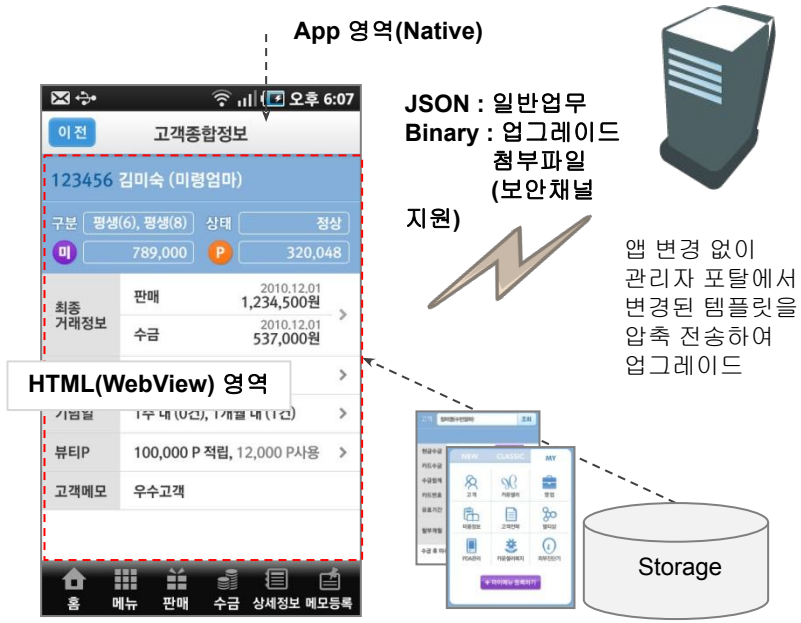
bizMOB™ Hybrid App은 HTML UI 템플릿을 단말의 안전한 저장소에 보관하며, 서버에서는 데이터만 가져 옵니다. 이를 통해 Transaction 속도를 향상시킬 수 있으며, 강력한 보안 및 버전 관리가 가능합니다.

일반적인 Hybrid Web – 웹캐싱 방식



bizMOB™ Hybrid App – 샌드박스 방식

1. UI 리소스를 제외한 최소의 데이터 Transaction 수행, 속도 향상
2. 전송되는 데이터에 대한 암호화 등 보안 처리 탑월
3. Client 소스는 앱실행시 강제 업데이트 다운로드로 최신버전 보장

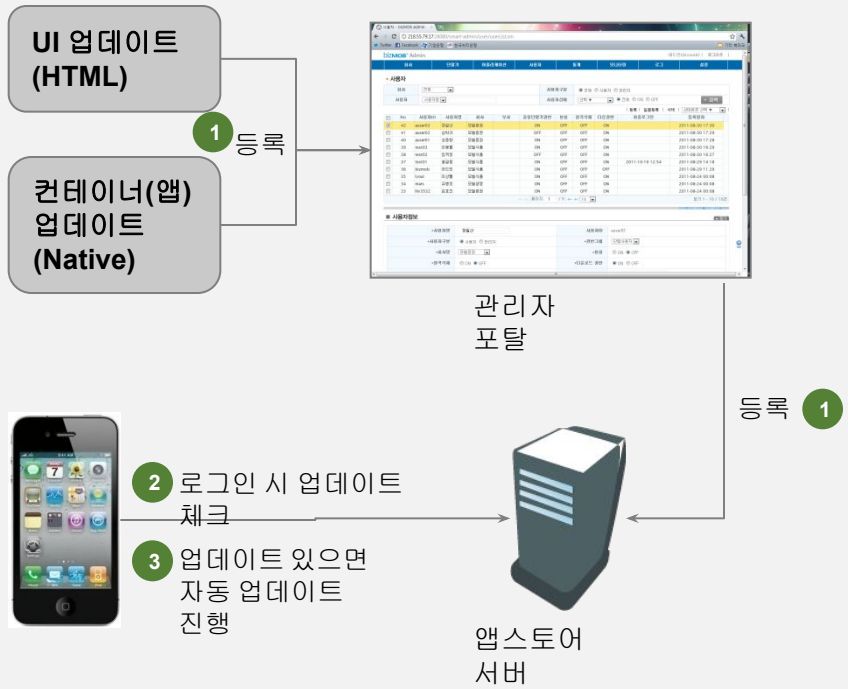


✔ 앱 배포 및 업데이트

bizMOB™ 하이브리드 앱의 배포는 관리자 포탈과 앱스토어 서버를 통해 이루어지며, 초기 배포 후에는 2단계로 분리된 자동 업데이트 메커니즘을 통해 픽스 및 업그레이드 패치가 이루어집니다. B2C용 앱 초기 배포는 Open Market(Google Play store와 apple appStore) 을 통해 이루어 집니다.

초기 앱 등록 및 배포

자동 업데이트



✔ bizMOB Platform 제품군 – Service Builder(개발 환경 제품)

Service Builder 는 클라이언트(모바일)와 서버간의 전문 설계/관리 툴로, 전문의 생성, 수정, 테스트 기능을 제공하며, 개발자간에 전문 공유를 쉽게 제공하여 변경된 전문을 바로 확인을 할 수 있도록 합니다.

구 분	특 징
JSON 타입 사용	웹에서 널리 쓰이는 JSON 타입은 파싱과 빌드를 위한 오픈 라이브러리를 사용할 수 있어, 어느 개발자나 쉽게 개발할 수 있음. 이를 통해 생산성을 크게 높임.
스키마 파일의 생성	JSON 스키마를 생성하여 서버와 클라이언트간에 주고 받는 데이터를 validation을 할 수 있도록 하여 전문의 안전성을 확보.
전문 테스트	생성한 전문을 테스트 하는 기능을 제공하여 전문에 대한 입력값과 결과값을 확인
사용자 편의성	개발자가 쉽게 사용 가능한 UI를 제공하여 빠르게 전문 생성

개발 생산성

개발자가 전문마다 중복으로 작성하는 소스 코드를 자동 생성

개발 용이성

하나의 서비스에 대해 클라이언트 화면UI에서부터 레거시 어댑터까지 서비스 흐름을 한눈에 확인

✔ bizMOB Platform 제품군 – Service Builder

< HE003 전문 >

- header
- body
  - subject
  - sender_name
  - sender_email
  - received_time
  - received_date
  - mail_uid
  - mail_to
    - email
    - display_name
  - mail_icon_type
  - mail_content_text
  - mail_content_html
  - mail_cc
    - email
    - display_name
  - mail_bcc
    - email
    - display_name
  - list
    - security_status
    - file_size
    - file_name
    - file_id
  - importance
  - attachment_flag

- header
- body
  - mail_uid

개별 POJO 클래스 생성

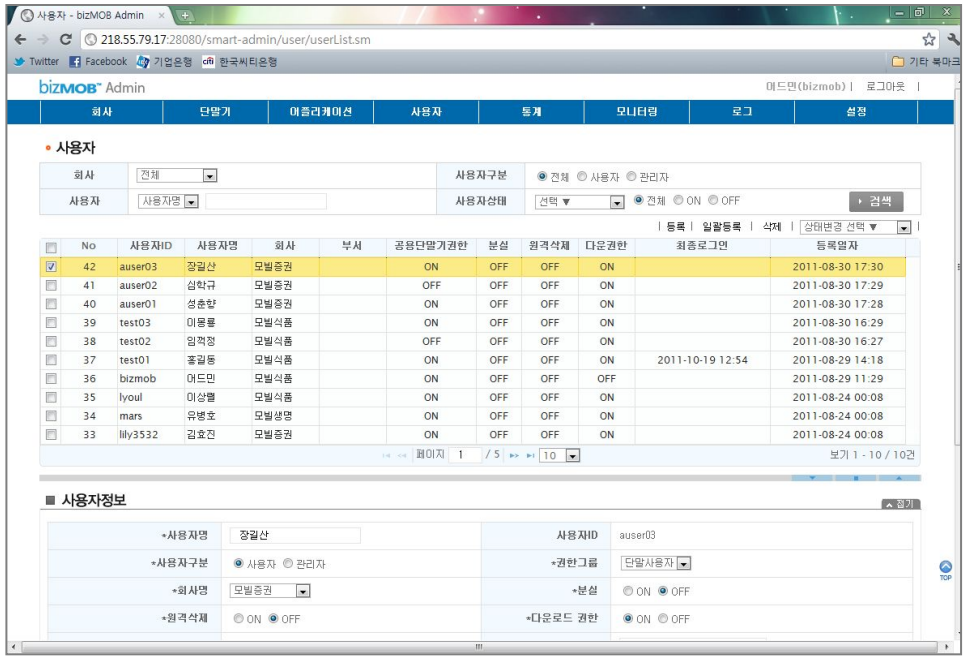
- project-model
  - src
    - com.mcnc.bizmob.project.model.HE0003
      - HE0003Request_Body.java
      - HE0003Request.java
      - HE0003Response_Body_List.java
      - HE0003Response_Body_Mailbcc.java
      - HE0003Response_Body_Mailcc.java
      - HE0003Response_Body_Mailto.java
      - HE0003Response_Body.java
      - HE0003Response.java
    - com.mcnc.bizmob.project.model.header
      - HWHeader.java
  - JRE System Library [J2SE-1.5]

전문1개에 Java 패키지 1개

```
this.$bizMOB.Network.requestTr({
  _bMock: false,
  _sTrcode: "HE0003",
  _oBody: {
    "mail_uid": "ex07@mcnc.co.kr"
  },
  _fCallback: function(res: any) {
    console.log(res);
    alert(JSON.stringify(res) );
  }
});
```

Client 전문 요청

✔ bizMOB Platform 제품군 – Admin Portal

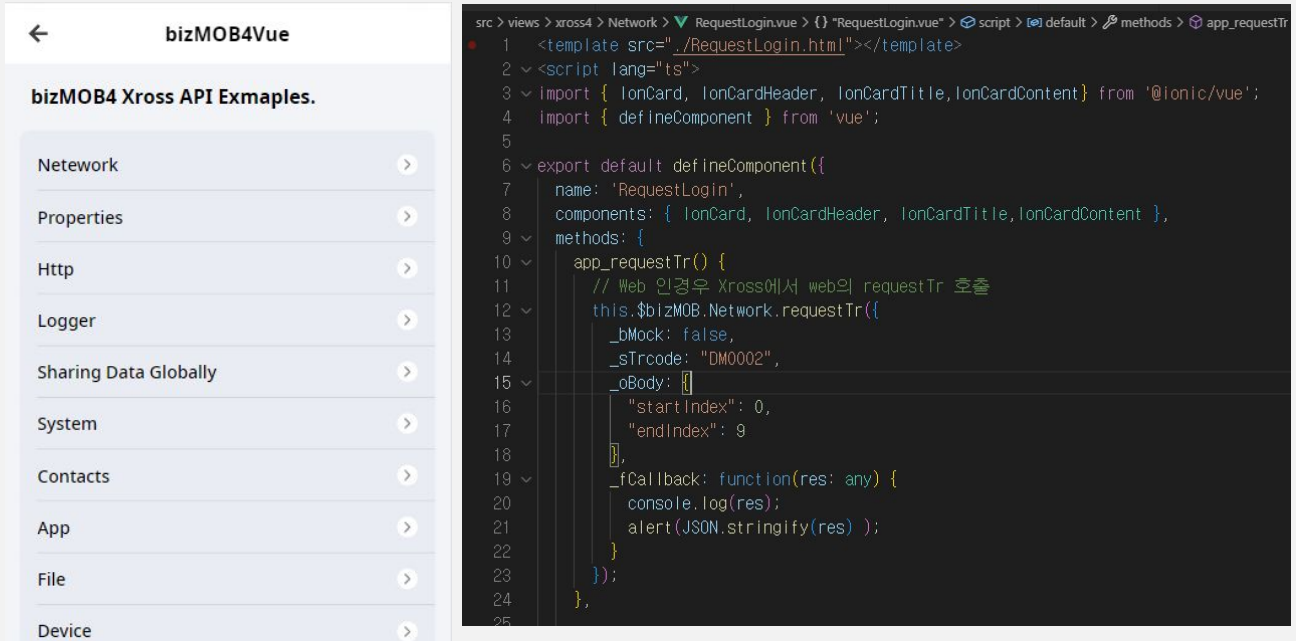


회사 관리	• 회사 리스트 • 회사 정보 • 회사 등록, 삭제
단말기 관리	• 지원 단말 리스트 • 지원 단말 등록, 삭제 • 지원 단말 세부 정보
어플리케이션 관리	• 기업용 어플리케이션 리스트 • 기업용 어플리케이션 등록, 삭제 • 기업용 어플리케이션 세부 정보
사용자 관리	• 사용자 리스트 • 사용자 등록, 삭제 • 사용자 세부 정보 • 등록 단말기 리스트
통계 관리	• 기간별 접속 현황 • 기간별 사용 현황 • 업무별 사용 현황
모니터링	• 서버 자원 모니터링 • 서버 상세 정보
로그 관리	• 포탈 로그 • 어플리케이션 사용 로그

✔ bizMOB Platform 제품군 – Xross 와 Core (Javascript API)

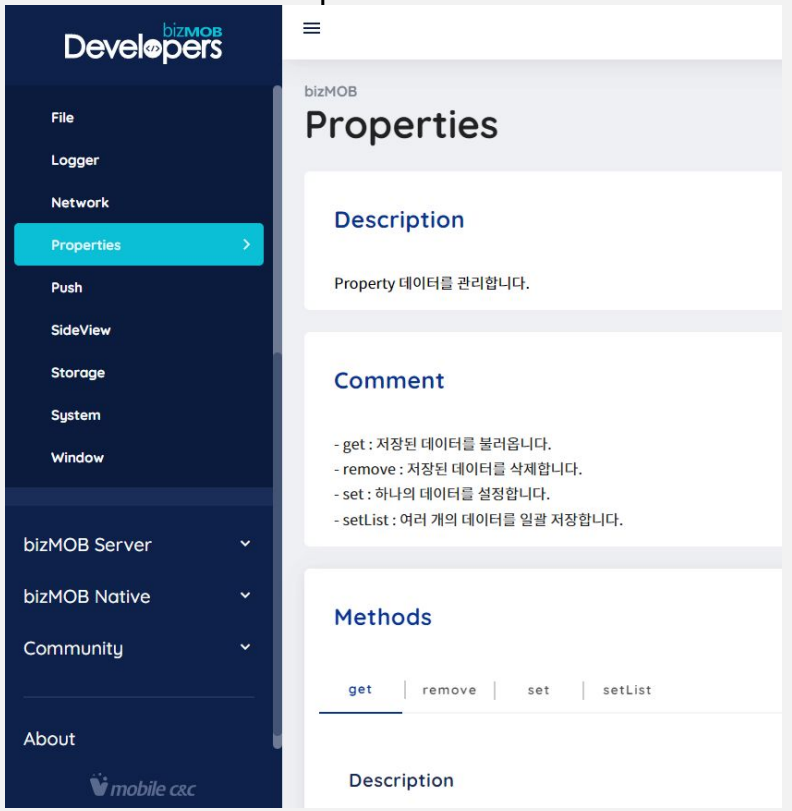
Native PlugIn 연동 개발 시에 Front 화면에서 API 호출에 의한 간편하게 Native기능을 구현 할 수 있습니다.
bizMOB™ Developer Site, Sample App을 통해 이슈, Forum, API 등 다양한 정보를 제공 하고 있습니다.

[Sample App Examples]




```
src > views > xross4 > Network > RequestLogin.vue > {} "RequestLogin.vue" > script > default > methods > app_requestTr
1  <template src="./RequestLogin.html"></template>
2  <script lang="ts">
3  import { IonCard, IonCardHeader, IonCardTitle,IonCardContent} from '@ionic/vue';
4  import { defineComponent } from 'vue';
5
6  export default defineComponent({
7    name: 'RequestLogin',
8    components: { IonCard, IonCardHeader, IonCardTitle,IonCardContent },
9    methods: {
10     app_requestTr() {
11       // Web 인경우 Xross에서 web의 requestTr 호출
12       this.$bizMOB.Network.requestTr({
13         _bMock: false,
14         _sTrcode: "DM0002",
15         _oBody: {
16           "startIndex": 0,
17           "endIndex": 9
18         },
19         _fCallback: function(res: any) {
20           console.log(res);
21           alert(JSON.stringify(res) );
22         }
23       });
24     },
25
```

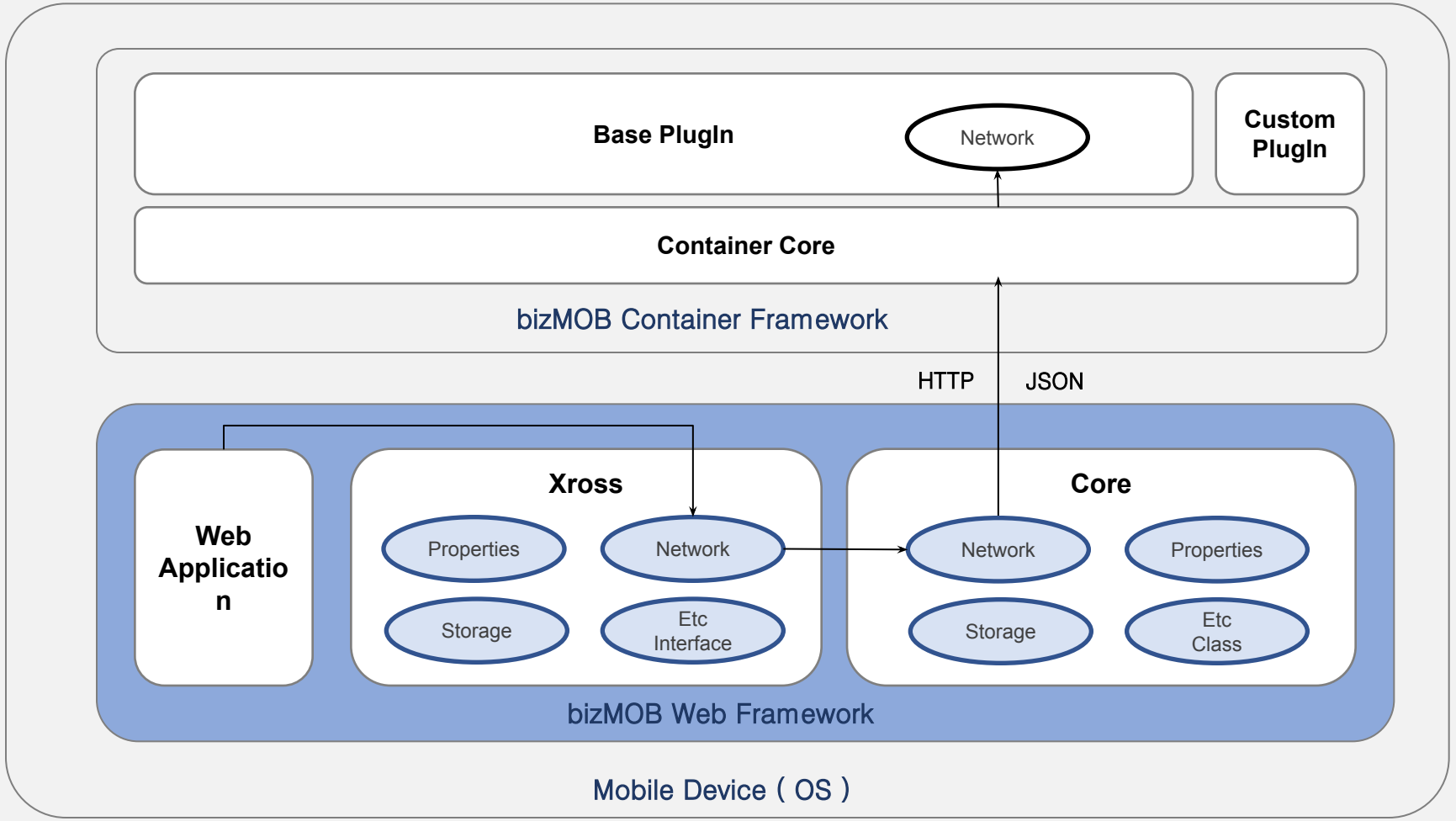
[API WebSite]



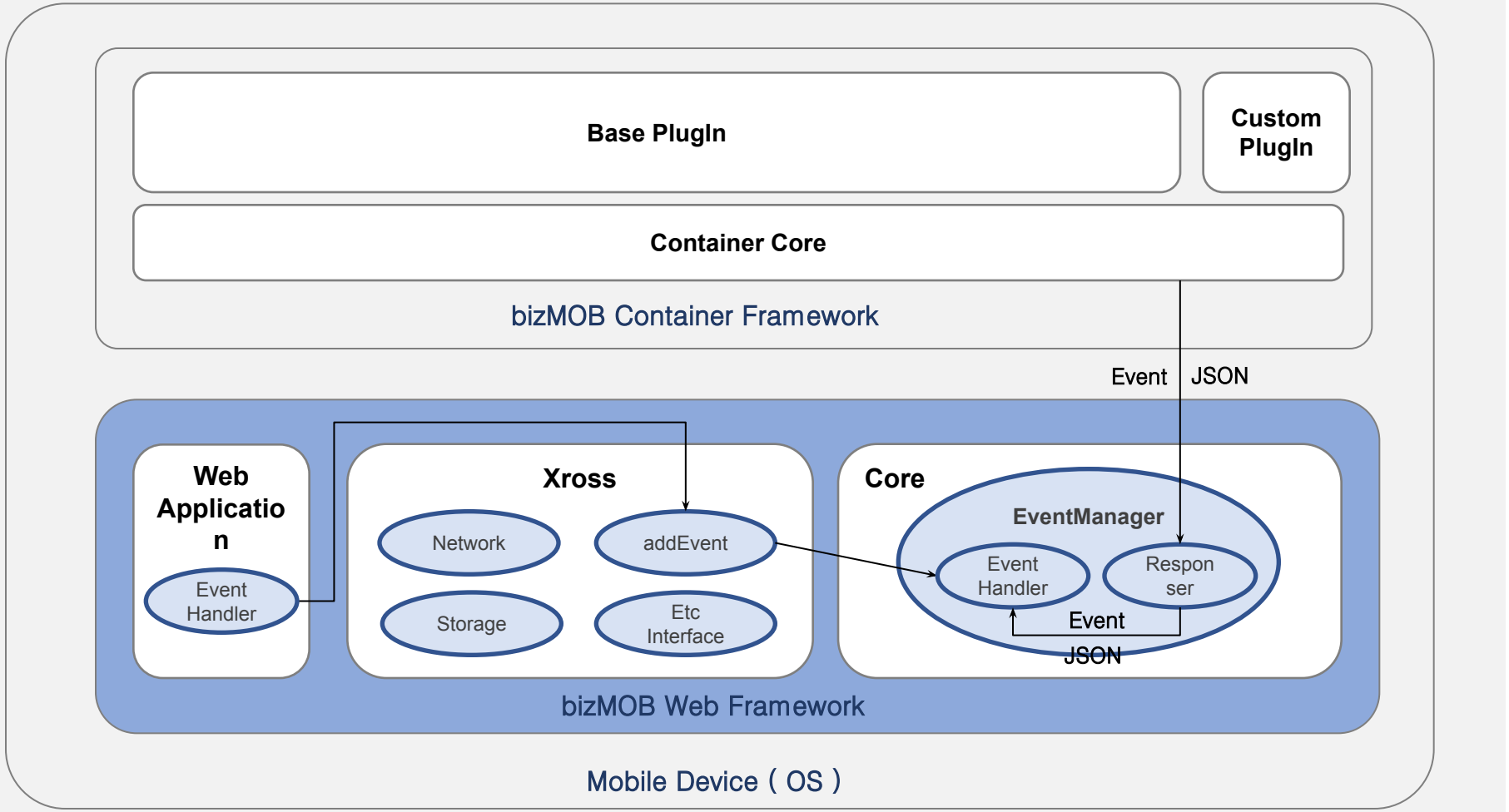
✔ bizMOB Platform 제품군 – bizMOB Xross 와 Core (Javascript API)

항목	설명	대표기능
App	App(bizMOB) 관련 API	앱 종료, 앱 타임아웃
Device	단말기 관련 API	단말기 정보
System	OS 제공 기능 API	전화, 문자, 지도
Contacts	연락처 관련 API	연락처목록 검색
Network	네트워크 관련 API	bizMOB 서버 통신
File	파일 관련 API	파일 정보, 복사, 삭제, 압축, 다운로드, 업로드
Properties	영구 저장소 관련 API	영구 데이터 저장, 삭제
Storage	휘발성 저장소 관련 API	휘발성 데이터 저장, 삭제
DataBase	컨테이너 SQLite DB 관련 API	DB 연결, CRUD
Logger	Web Application 로깅 관련 기능	로그 기록

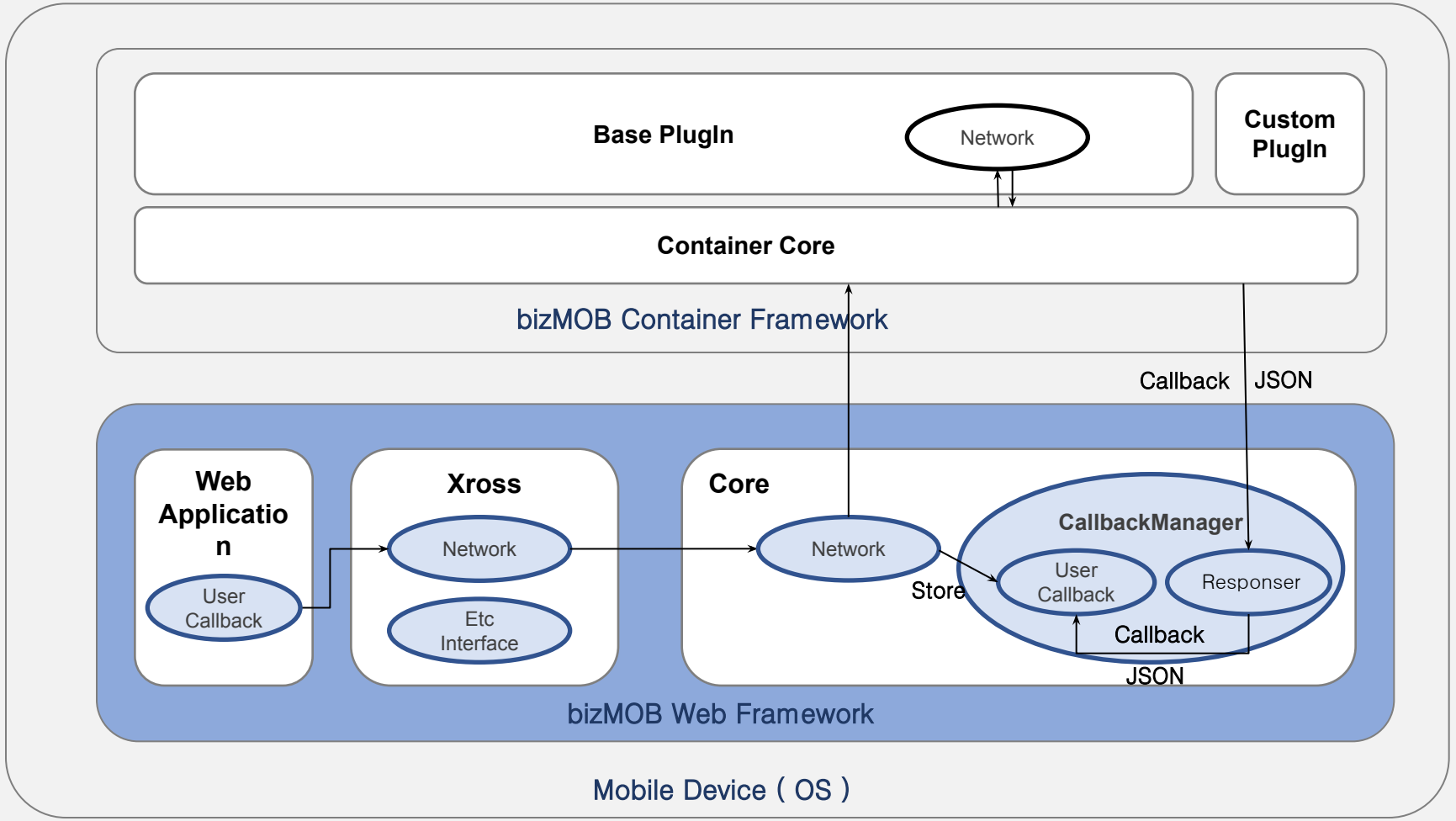
✔ bizMOB Client Architecture – Xross, Core



✔ bizMOB Client Architecture – Event

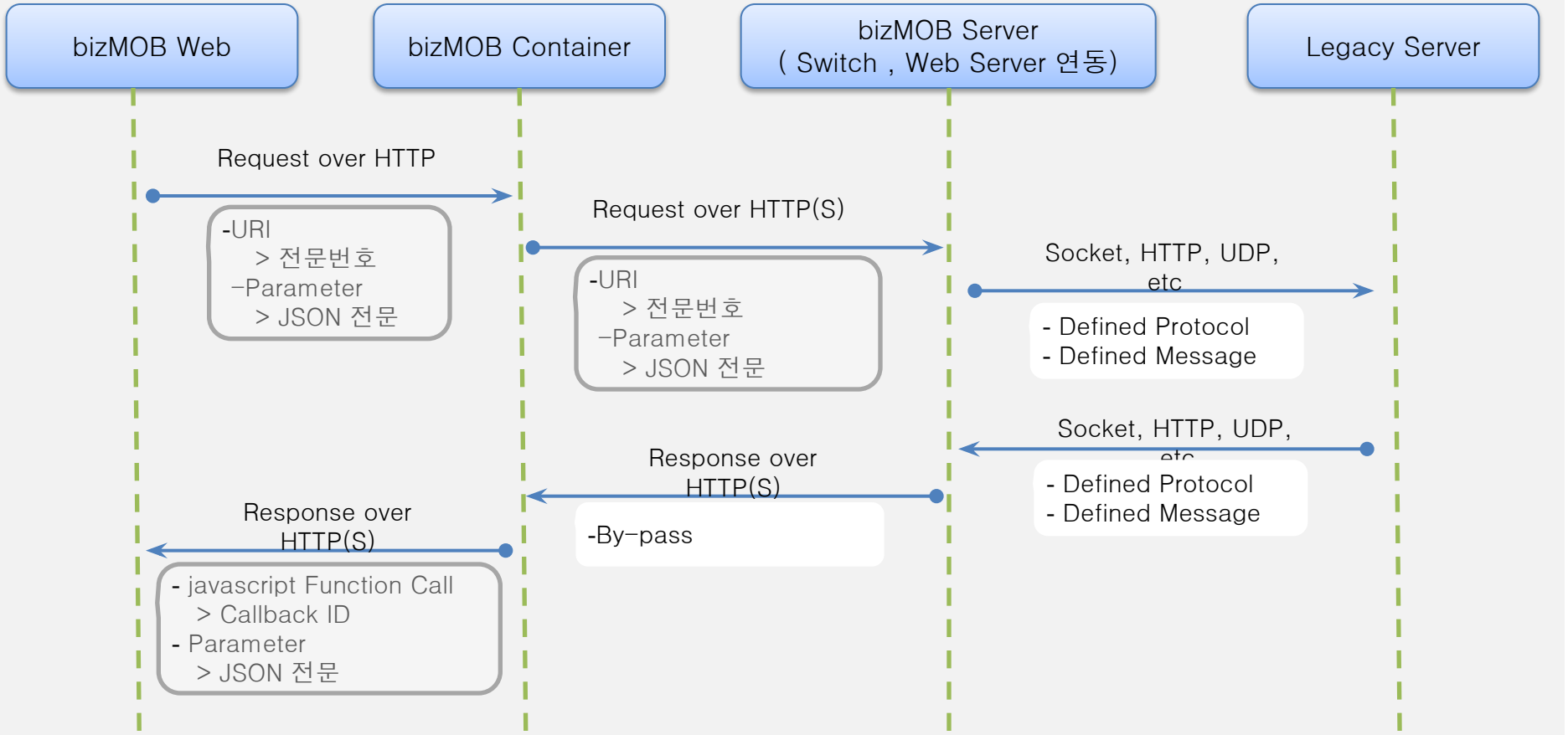


✔ bizMOB Client Architecture – Callback



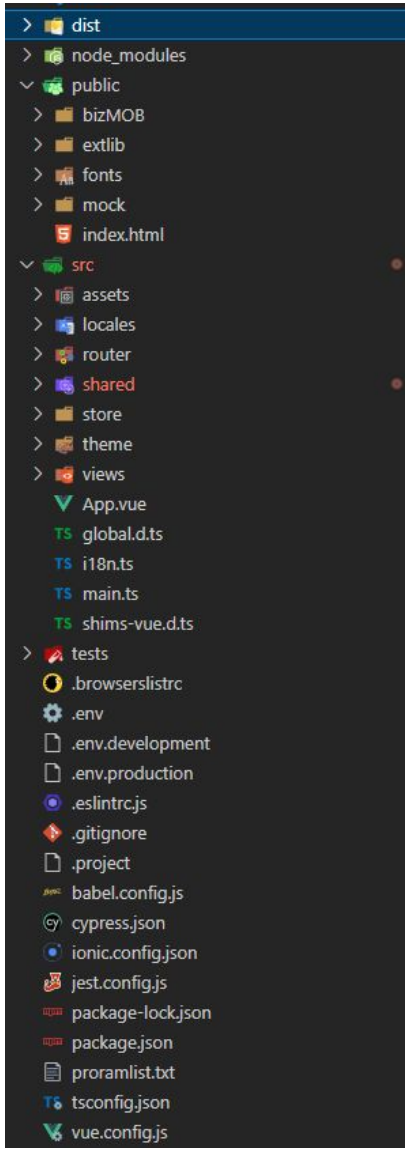
✔ bizMOB Client Architecture –

모바일 클라이언트 <-> 모바일 서버
- **JSON 전문 over HTTP**
- 보안을 강화하기 위해서 **SSL**을 적용한 **HTTPS** 사용 가능



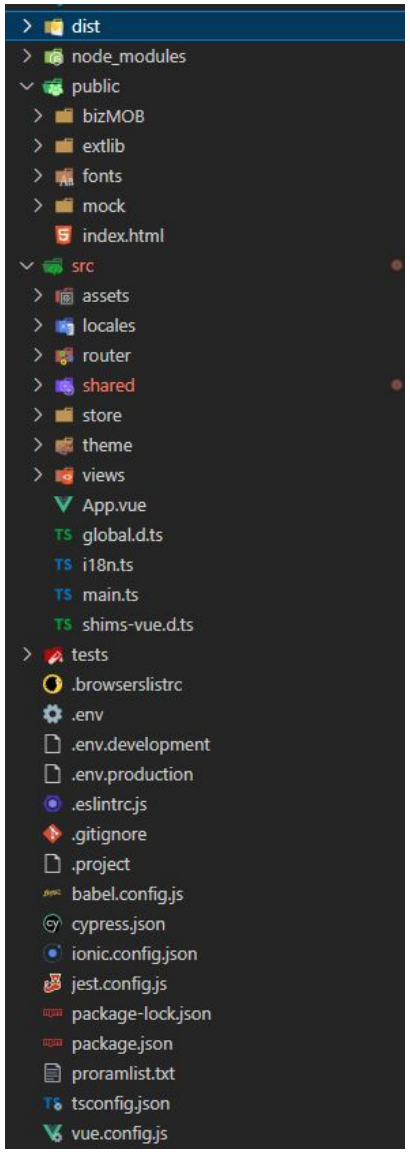
bizMOB4 신규/변경 기능

프로젝트 폴더 구성



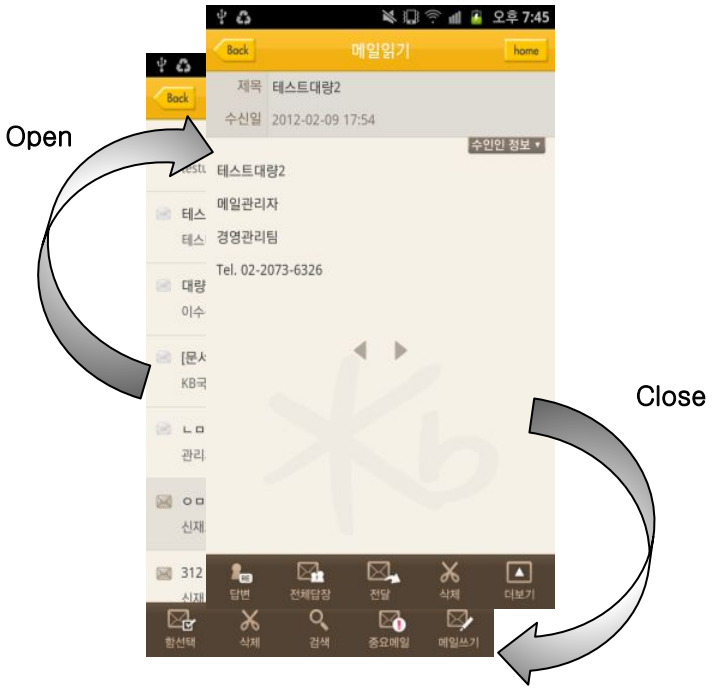
폴더/파일 명	Description
/dist	프로젝트 배포시에 빌드된 컨텐츠 파일이 생성되는 폴더. 해당 폴더 아래 환경설정 이름을 가진 폴더가 생성되고 그 하위에 배포 컨텐츠가 생성된다.
/node_module	Package.json에 정의된 설치한 모든 플러그인 파일들이 생성된다. 주의) 용량이 매우 크므로 svn에 저장하지 않는다.
/public	정적인 Resource를 저장하는 공간으로 폰트, 외부JS lib와 같은 것들만 저장한다.
/public/bizMOB	Native API script lib 주의) 수정 금지.
/public/mock	서버 전문이 정의되고 서버 개발이 완료되기 전까지 사용하기 위한 mock데이터를 정의한 json파일 폴더
/public/extlib	외부 javascript lib 파일 저장 폴더
/public/fonts	특정 폰트 사용시 저장 폴더. Default : Google NotoSans Font
Index.html	Welcome 파일
/src	프로젝트 소스 폴더
/src/assets	프로젝트 리소스파일 저장 폴더 css, image 파일들을 저장한다.
/src/locales	다국어 설정시 언어팩 저장 폴더
/src/router	View Rounting 설정 폴더
/src/shared	프로젝트 공통 기능 class 정의폴더. bizMOB, constants(상수선언), project(프로젝트 공통) 으로 분류 주의) bizMOB 폴더내 파일은 수정 금지.
/src/store	공유 데이터 사용 Vuex plugin 폴더 주의) 수정 금지.
/src/views	View Screen 저장 폴더

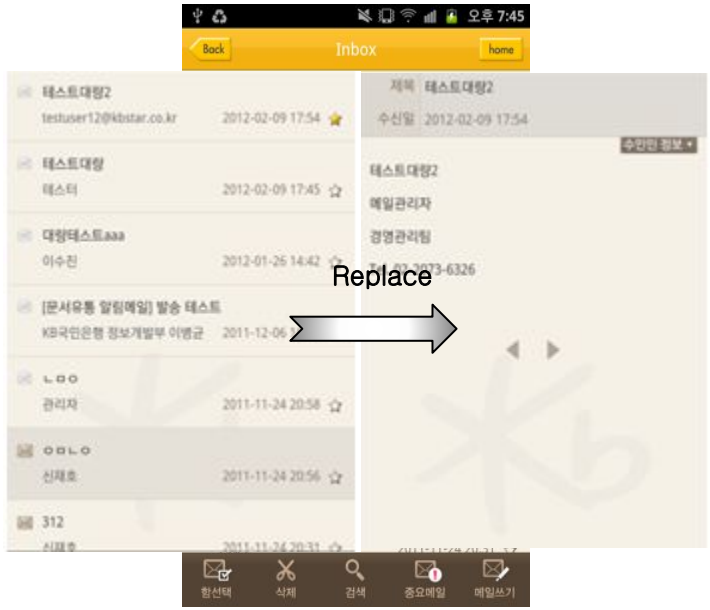
프로젝트 폴더 구성



폴더/파일 명	Description
/src/App.vue	Bootstraping vue 파일. 주의) 어플리케이션 초기화 관련 로직 외에는 코딩하지 않는다.
/src/main.ts	Bootstraping Class 파일. 주의) 어플리케이션 초기화 관련 로직 외에는 코딩하지 않는다.
/env /env.xxxxxxx	어플리케이션 환경변수 정의 파일
/package.json /package-lock.json	설치한 npm 플러그인 정보 파일 주의) 플러그인 설치시 cli를 통해서만 설치/삭제 한다.
/vue.config.js	Vue framework 설정 파일

Vue Router Class를 활용하여 화면을 제어하며 bizMOB3.x 의 WebView 스택이 중첩되는 구조가 아님.





새로운 화면을 생성

- 이전 화면의 상태를 보존이 필요할 경우 (화면 목록)
- 일반적인 게시판 목록/상세 구조 개발 시 사용

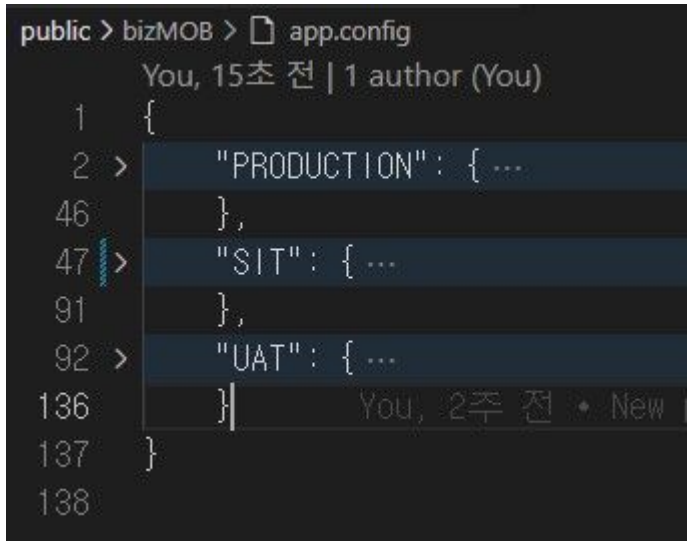
Sample Project (Vue3 – Navigator 참조)
e.g) this.$route.push("/bbs/detail");

다른 화면으로 이동

- History Back시 이전 화면으로 돌아가기
- 화면의 스텝이 단계별로 진행되는 경우 사용 (단계별 화면 이동 설계시)

Sample Project (Vue3 – Navigator 참조)
e.g) this.$route.replace("/bbs/step2");

app.config – 앱 구동 설정, 환경별로 설정 추가 가능



/public/bizMOB/app.config

- 기본적으로 Production(운영), UAT(staging), SIT(개발환경)을 제공한다.
- 추가 환경설정이 필요할 경우 Native팀과 협의하여 추가 가능하다.

Depth 1	Depth 2	Depth 3	Description	설정값
VERSION	NATIVE		앱 버전 정보	4.0 (임의 설정 금지)
	FRONT		프론트 버전 정보	4.0 (임의 설정 금지)
SETTINGS	TYPE		앱 타입	B2B , B2C
	APPLICATION_TYPE		앱 모드	MOBILE , WEB
	NOTICE_USE		공지기능 사용 여부	true, false
SECURITY	CONTENTS_INTEGRITY		Contents 위변조검사 실행 여부	true, false
	INACTIVITY_SECONDS		앱 미사용 시간 설정(초단위)	0 (Unlimited), 1 이상 정수
STATUSBAR	MODE		표시 모드 설정	DEFAULT , EXTEND , FULL
	BGCOLOR		배경색	RGB 값.
	TEXT_BLIGHT		야간모드 설정	BLACK/WHITE
WEBVIEW	BACKGROUNDCOLOR	NORMAL	웹뷰 배경색	aRGB 값.
		DIALOG	팝업 창 배경색	aRGB 값.
NETWORK	FILE	CONNECT	파일 업로드/다운로드시 서버 Connection Timeout 시간설정(초단위)	1 이상 정수
		READ	파일 업로드/다운로드시 서버 ReadTimeout시간설정(초단위)	1 이상 정수
	API	CONNECT	전문 통신시서버 Connection Timeout시간설정(초단위)	1 이상 정수
		READ	전문 통신시 서버 ReadTimeout시간설정(초단위)	1 이상 정수
PUSH	URL		bizPush 서버 URL	URL String

앱 설정 세부항목

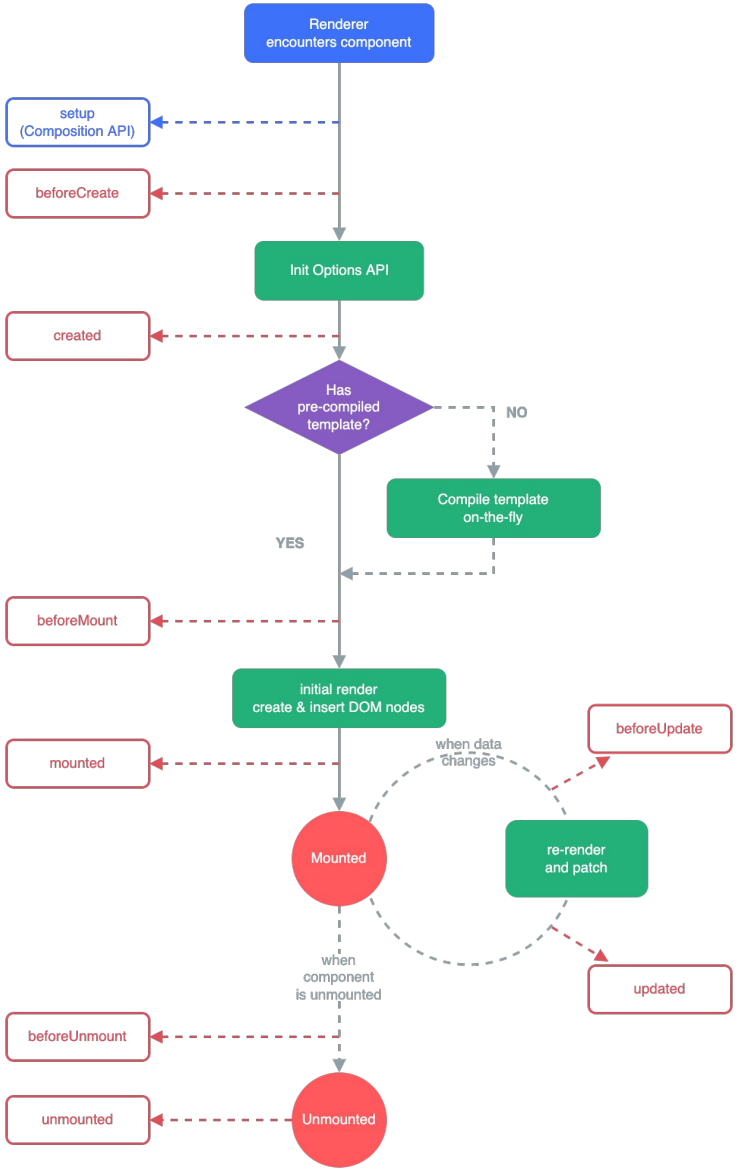
II. LifeCycle Hooking

개요

일반 웹 파일이 렌더링 되는 과정처럼 Vue프레임워크에서는각 Vue Component들이 생성/삭제 될때 마다 각 과정별로 Event Handler 인터페이스를 제공한다. Component 의 전체 lifecycle은 오른쪽 그림과 같다.

Description

Hooking Method	Description
setup()	렌더링 엔진이 Component를 로드 했을때 발생하는 이벤트. Composition API인 setup 메소드로 정의 할 수 있다.
beforeCreate()	렌더링 엔진이 Component instance 생성/초기화 한 직후에 발생하는 이벤트.
created()	Component instance의 전체 Options API를 실행한 직후에 호출되는 이벤트
beforeMount()	생성된 Component instance를 DOM에 삽입 직전에 호출되는 이벤트
mounted()	DOM에 로드 후 호출되는 이벤트. Window Object의 load 이벤트와 동일하다.
beforeUpdate()	Component내 Re-rendering/painting 되기 직전에 호출되는 이벤트
update()	Component내 Re-rendering/painting 된 후에 호출되는 이벤트
BeforeUnmount()	Component가 DOM에서 삭제되기 전에 발생하는 이벤트
unmounted()	Component가 DOM에서 삭제 된 후 발생하는 이벤트



코딩 가이드

준수사항

bizMOB에서 javascript Library 수정 금지 (Xross, Core)
/public/bizMOB 하위 javascript Library에 대한 재정의 금지, 개선사항 및 버그가 존재하는 경우 담당자에게 수정을 요청

shared 폴더내 Service Classes 수정 금지
/src/shared/bizMOB 하위 typescript services에 대한 수정 금지. 개선사항 및 버그가 존재하는 경우 담당자에게 수정을 요청

변수, 함수 명명
의미있는 이름으로 부여
변수 및 함수는 Camel Casing(소문자를 기본으로, 구분되는 단어의 첫글자를 대문자로) 규약 적용
올바른 예) userName(사용자명), orderDate(주문일)
잘못된 예) a, num1, Value, tmp
상수는 대문자를 기본으로 하고 구분되는 단어 사이에 '_' 삽입
ex) GRADE_A(A등급)
bizMOB 전문의 request는 'req' + 전문코드 선언
bizMOB 전문의 response는 'res' + 전문코드 선언

Code Format
제어문(if, for 등)에서 {} 생략없이 사용하고 {}는 줄바꿈을 사용하여 코딩
String을 선언할 경우 ' 대신 " 사용
조건문/반복문에 사용되는 비교값은 하드코딩하지 않고 상수로 선언하여 사용
상수 변수는 대문자로 단어의 구분은 "_"를 사용하고 Page변수내 선언하거나 공통파일(ex : common.js)을 생성하여 관리.

준수사항

조건/반복문

분기하는 사유 또는 조건을 설명. 비교값은 변수명을 의미있게 부여하여 사용한다.

```
1.      //Correct Case
2.      // 선택된 제품이 사용자가 선택할 수 있는 제한 갯수를 넘을 경우
3.      if(productCheckList.length > bizMOB.Constant.MAXIMUM_COUNT)
4.      {
5.      }
```

```
1.      // Bad Case
2.      // 선택된 제품이 사용자가 선택할 수 있는 제한 갯수를 넘을 경우
3.      if(productCheckList.length > 10 )
4.      {
5.      }
```

Static Resource

정적인 리소스들은 반드시 public 폴더 하위에 두어 사용합니다.

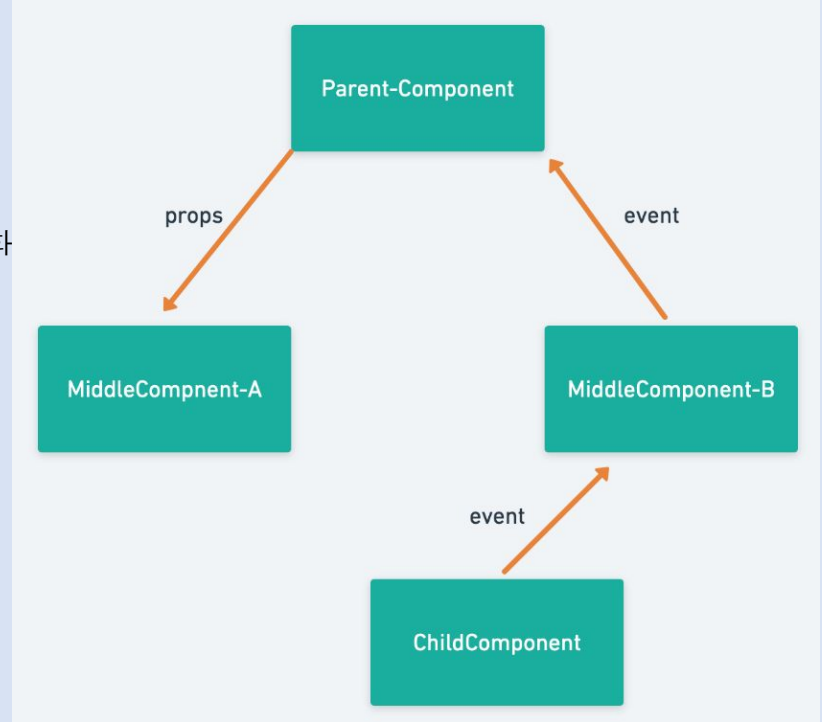
> bizMOB : bizMOB Framework 관련 파일들
> extLib : 외부 JS library 를 위한 폴더
> fonts : 앱에서 사용할 font 파일. 기본 폰트로 Google NotoSans 폰트가 있다. (다국어 프로젝트일 경우는 필수)
> mock : 서버 전문 통신시 mock data를 활용하기 위한 response json 설정 파일 폴더

Vuex를 사용하는 이유

기본적으로 vue 생태계에서 컴포넌트 간 여러 정보들을 관리하려면 부모-자식 관계로 데이터들을 넘겨주고 받아야한다. 하지만 현실적으로 복잡한 컴포넌트 관계속에서 계속 주고 받으려면 여러 컴포넌트 파일에서 data 속성을 관리해주어야 하는 단점이 있다.
따라서 여러 컴포넌트에서 공유해야할 데이터가 있다면 Vuex를 활용하여 데이터를 주고 받으면 편리하다.

State VS WebStorage

Web 어플리케이션 개발시 대표적인 데이터 공유 Storage 기능으로 브라우져의 Web Storage 기능이 있다.
스토리지는 Local/Session 두가지 종류의 스토리지 기능을 제공 하지만 몇가지 차이점을 이해 해야한다.



Vuex(state)
- LifeCycle : State는 어플리케이션 종료시(Framework Reload 또는 브라우져 새로 고침) 데이터가 초기화
- Data 공유 : 실시간으로 Component간에 공유된다.
- 적용대상 : Application 전체 및 여러 Component끼리 실시간으로 공유되어야 하며 Application 시작시 데이터 갱신이 필요한 경우 e.g) 로그인 후 사용자 정보, 사용자별 메뉴 권한

LocalStorage
- LifeCycle : 영구.
- Data 공유 : 각 Component에서 매번 데이터를 호출하여 갱신하여야 한다..
- 적용대상 : Application 전체 및 고정 데이터로서 여러 Component에서 공유할 데이터가 필요한 경우

SessionStorage
- LifeCycle : 브라우져 종료시
- Data 공유 : 각 Component에서 매번 데이터를 호출하여 갱신하여야 한다..
- 적용대상 : Application 전체 및 고정 데이터로서 여러 Component에서 공유할 데이터가 필요한 경우

Service로 사용하기
bizMOB GlobalSharedService 를 import 합니다. setup composition API내 서비스를 생성합니다.

```
import GlobalSharedService from “@/shared/bizMOB/GlobalDataService”;

setup(){
   const { getGlobalDataByKey,removeGlobalDataByKey, setGlobalDataByKey } = GlobalSharedService();

   return {   getGlobalDataByKey,removeGlobalDataByKey, setGlobalDataByKey   }
}
```

제공된 mutations를 사용하여 데이터를 사용합니다. 저장할 데이터 Type은 원하는 형태로 지정시 자동으로 저장됩니다.

```
mounted(){
   this.setGlobalDataByKey("mydata1","Composition String");
   this.setGlobalDataByKey("mydata2",{data1:"Composition JSON"} );
},
```

Class로 사용하기
bizMOB GlobalDataUtil Class를 import 합니다. setup composition API내 인스턴스를 생성합니다

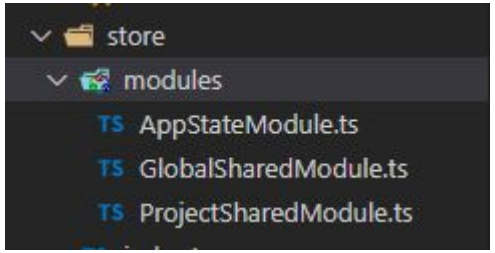
```
import GlobalDataUtil from '@/shared/bizMOB/GlobalDataUtil';

 setup() {
   const gdu = new GlobalDataUtil();

   return { gdu }
  },
```

제공된 methods를 사용하여 데이터를 사용합니다. 저장할 데이터 Type은 원하는 형태로 지정시 자동으로 저장됩니다.

```
 this.gdu.setGlobalDataByKey("Function1","Function String" );
 this.gdu.setGlobalDataByKey("Function2",{data1:"Function JSON"} );
```



1. Custom state 사용을 원할 경우 ProjectSharedModule.ts를 수정하여 직접 구현할 수 있습니다.
2.별도 모듈 생성시에는 등록과정이 필요합니다.

mock data 정의

프로젝트 초기 Server 전문이 미 완성일 경우 Service Builder에 정의된 내용을 기반으로 bizMOB 전문 프로토콜의 Response Object를 정의하여 사용한다.

mock data file 생성

```
1.  // 기본 포맷
2.  {
3.     "header": {
4.    "result": true,
5.    "error_code": "",
6.    "error_text": "",
7.    "trcode": " HAL0010", // Service Builder에 정의된 TrCode
8.    "trcodeName": ""
9.     },
10.    "body": {
11.          // Service Builder에 정의된 내용 작성
12.    }
13. }
```

component 내 사용 예시

```
1.  this.$bizMOB.Network.requestTr({
2.        _bMock: true ,  // true로 설정시 정의한 mock데이터를
    리턴
3.        _sTrcode: "HAL0010",
4.        _oBody: {
5.              // Service Builder에 정의된 내용 작성
6.         },
7.        _fCallback: (res: any) => {
8.           // json에 정의된 mock data가 parameter로 전달
9.         }
10. });
```

개요

컴포넌트를 생성하기 위한 html, vue 파일 기본 설정을 따라 생성한다.

Component 폴더 생성
프로젝트 내 메뉴 또는 기능에 맞게 폴더를 생성하여 하위에 Templet, Vue파일을 생성한다. 파일명은 UX설계서에 정의된 화면 ID를 기준으로 생성한다.
(그림 1-1)

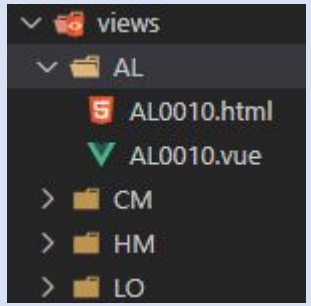


그림1-1

Templet 파일 생성
퍼블리싱 된 html파일을 복사하여 Templet파일에 붙여 놓고 저장한다. (그림 1-2)

Vue 파일 생성
Vue 파일을 생성하고 Component에서 사용할 Templet 파일을 지정한다. (그림 1-3)

```
<template src="./introPage.html"></template>
<!-- <style src="./style.css"></style> -->
<script lang="ts">
import { IonCard, IonCardContent, IonCardHeader, IonCardTitle } from '@ionic/vue';
import { defineComponent } from 'vue';

export default defineComponent({
  name: 'IntroPage',
  components: { IonCard, IonCardHeader, IonCardTitle, IonCardContent },
  data() {
    return {
      appMode: process.env.VUE_APP_MODE
    }
  },
```

그림 1-3(Intro.vue)

```
<ion-page>
    <ion-header :translucent="false">
        <ion-toolbar>
            <ion-title>bizMOB4Vue</ion-title>
        </ion-toolbar>
    </ion-header>

    <ion-content :fullscreen="true">
        <h2 class="page_tit">{{ $t("intro.title") }} ({{ appMode }})</h2>
        <ion-list>
            <ion-item detail v-on:click="goVue3List"><ion-label>Vue3</ion-label></ion-item>
            <ion-item detail v-on:click="$router.push({ name: 'Xross4' })"><ion-label>Xross4 2</ion-label></ion-item>
            <ion-item detail v-on:click="$router.push({ name: 'Native' })"><ion-label>Native</ion-label></ion-item>
        </ion-list>
    </ion-content>
</ion-page>
```

그림 1-2 (Intro.html)

개요

각 컴포넌트를 호출하기 위한 라우팅 정보를 세팅한다.

router/index.ts

라우터 클래스를 로딩하여 선언할 수 있는 최상위 Root Router class이며 프로젝트에서 생성한 라우팅 클래스를 import하여 설정 할 수 있다. (그림 1-1 그림 참조)

router/xxxx.ts (Routes Class)

라우터 폴더 내에 생성하고 모든 컴포넌트에 대하여 라우팅 정보를 생성한다.
선언시 아래 Properties를 사용 하여 정의 한다. (그림 1-2 그림 참조)

Properies	Description
path	컴포넌트 호출시 URI로 호출할 경우에 대한 path 설정을 지정한다. 기본값은 "/"(root)로 설정되어 있다.
component	Path 설정과 매핑할 Component을 import한 후 class명을 설정한다.
redirectTo	특정 uri를 redirect 시키고 싶을때 설정한다.

```
import { createRouter, createWebHistory } from '@ionic/vue-router';
import { RouteRecordRaw } from 'vue-router';
import RootRouter from './RootRouter';
const routes: Array<RouteRecordRaw> = ([] as Array<RouteRecordRaw>).concat(RootRouter);

const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
})
```

그림1-1

```
import IntroPage from '../views/TLO/TLO0010.vue';

export default [
    {
        path: '/',
        redirect: '/intro'
    },
    {
        path: '/intro',
        name: 'RootRouter',
        component: IntroPage
    },
```

그림 1-2

개요

컴포넌트간 데이터 공유를 위한 글로벌 공유 데이터 설정하기

Service Import

Global Data 설정을 위한 Service Class를 import합니다.

```
import GlobalSharedService from '@/shared/bizMOB/GlobalDataService';
```

setup API에서 instance를 생성한다.

```
const { getGlobalDataByKey,removeGlobalDataByKey, setGlobalDataByKey } = GlobalSharedService();
return { getGlobalDataByKey, setGlobalDataByKey, removeGlobalDataByKey, gdu }
```

bizMOB API

데이터 저장/삭제/조회는 API에서 제공하는 메서드를 통해 사용합니다.

Methods	Description
setGlobalDataByKey	데이터 저장, 데이터는 Script에서 제공하는 기본 데이터 타입으로 Key, Value 를 인자로 사용한다. (그림 1-1 참조)
getGlobalDataByKey	데이터 조회, 저장시 지정한 Key값으로 조회한다. (그림 1-1 참조)
removeGlobalDataByKey	데이터 삭제, 저장시 지정한 Key값으로 삭제한다. (그림 1-1 참조)

```
import GlobalSharedService from '@/shared/bizMOB/GlobalDataService';
import GlobalDataUtil from '@/shared/bizMOB/GlobalDataUtil';

import { IonCard, IonCardContent, IonCardHeader, IonCardTitle } from '@ionic/vue';

import { defineComponent } from 'vue';

export default defineComponent({
  name: 'GlobalData',
  components: { IonCard, IonCardHeader, IonCardTitle,IonCardContent },
  setup() {
    const gdu = new GlobalDataUtil();

    const { getGlobalDataByKey,removeGlobalDataByKey, setGlobalDataByKey } = GlobalSharedService();
    return { getGlobalDataByKey, setGlobalDataByKey, removeGlobalDataByKey, gdu }
  },
  mounted(){
    this.setGlobalDataByKey("mydata1","Composition String");
    this.setGlobalDataByKey("mydata2",{data1:"Composition JSON"} );
    this.removeGlobalDataByKey("mydata1");
    const CompositionData =  this.getGlobalDataByKey("mydata2");
    this.$logger.log(CompositionData);
  },
```

그림 1-1

개요

다국어 지원은 i18n Module을 사용합니다.

참조)
https://vue-i18n.intlify.dev/

Description

지원 언어팩 생성하기
/src/locale/ 폴더 하위에 lang cd값(ISO 639-1 codes 2digit)을 이름으로 하여 파일을 생성합니다.

언어팩 설정
생성한 언어팩 파일 내에 Project 규칙에 맞게 json 형태로 데이터를 작성합니다.

```
{
  "intro": {
    "title": "bizMOB sample project"
  },
    "main": {
      "greeting": "Wellcome",
      "name_greeting": "Hello {name}"
    }
}
```

모듈 문법 활용
https://vue-i18n.intlify.dev/api/general.html#createi18n

{lang code}.json

개요

다국어 프로젝트의 경우 서비스를 별도로 등록해야 동작을 하며 프로젝트 기본값은 서비스가 시작되지 않습니다.

Service import
bizMOB LocalService Class를 import 합니다. setup Composition API를 활용하여 서비스를 생성합니다.

```
setup(){
   const localSvc = new LocaleService();

   return {   localSvc   }
}
```

Service 등록
App.vue 파일을 열고 LocalService를 시작합니다. mounted Composition API를 활용하여 서비스를 등록합니다.

```
mounted(){
   this.localSvc.initLocale();
},
```

언어변경 설정
사용자/어플리케이션내에서언어변경시 감지를 위한 이벤트 및 LocalService의 이벤트 핸들러를 등록해 줍니다.

```
 watch: {
   '$i18n.locale': function(newLocaleCd) {
     this.localSvc.changeLocale(newLocaleCd);
   },
 },
```

```
import LocaleService from './shared/bizMOB/LocaleService';

export default defineComponent({
  name: 'App',
  components: {
    IonApp,
    IonRouterOutlet
  },
  setup(){
    const localSvc = new LocaleService();

    return {
      localSvc
    }
  },
  mounted(){
    this.localSvc.initLocale();
  },
  watch: {
    '$i18n.locale': function(newLocaleCd) {
      this.localSvc.changeLocale(newLocaleCd);
    },

  },
```

개요

어플리케이션 퍼포먼스를 위해서 로그 레벨을 환경별로 자동으로 조정되는 별도 Log Class를 사용합니다.

logger Class
Global 사용등록된 로거 클래스를 바로 사용할 수 있습니다.
this.$logger

Methods
Log level에 맞게 각 로그 메소드를 사용합니다.

Methods	Description
log	개발시 필요한 정보를 표시합니다.
Info	구간별 데이터 정보를 표시합니다.
warn	예외처리에 대한 별도 로그를 표시합니다.
error	에러 발생시 로그를 표시합니다.

```
this.$logger.log("this is log level.");
this.$logger.info("this is info level.");
this.$logger.warn("this is warn level.");
this.$logger.error("this is error level.");
```

bizMOB 신규/ 변경 Native PlugIn

개요

전체 네이티브 플러그인 사용 API정보는 아래 사이트를 참조 하십시오.
https://developers.mcnc.co.kr/

Class

`bizMOB.Localization`
- 다국어 정보 처리

Description

Methods	Options	return	Description
setLocale(options)	_fCallback : 콜백 함수	Void	현재 언어 정보를 저장합니다.
getLocale(options)	_sLocaleCd : 설정 언어코드 _fCallback : 콜백 함수	{ locale : 설정값 }	현재 설정된 언어정보를 가져옵니다.

Class

`bizMOB.Network`
- bizMOB Server 통신 플러그인

Description

Methods	Options	return	Description
changeLocale(options)	_sLocaleCd : 설정 언어코드	Void	통신시 요청할 언어 정보를 설정합니다.

requestLogin (options)	_bMock : mock data 사용여부
	_sUserId : 인증 받을 사용자 아이디입니다.
	_sPassword : 인증 받을 사용자 패스워드입니다.
	_sTrcode : 전문코드입니다.
	_oHeader : 전문 Header 객체입니다.
	_oBody : bizMOB전문 Body 객체입니다.
	_bProgressEnable(default:true) : 서버에 요청 통신 중일때 화면에 progress 를 표시할지에 대한 여부입니다.
	_fCallback : 서버와 통신 후 실행될 callback 함수입니다.
		{
		"header" : Object(전문 통신의 Header),
		"body" : Object(전문 통신의 Body)
		}
			bizMOB 서버에 사용자 정보 인증을 요청합니다.

requestTr (options)	_bMock : mock data 사용여부
	_sTrcode : 전문코드입니다.
	_oHeader : 전문 Header 객체입니다.
	_oBody : bizMOB전문 Body 객체입니다.
	_bProgressEnable(default:true) : 서버에 요청 통신 중일때 화면에 progress 를 표시할지에 대한 여부입니다.
	_fCallback : 서버와 통신 후 실행될 callback 함수입니다.
		{
		"header" : Object(전문 통신의 Header),
		"body" : Object(전문 통신의 Body)
		}
			bizMOB 서버와 전문 통신을 합니다.
			(post방식)

Class

`bizMOB.Push`
- bizMOB Push Server 통신 플러그인

Description

Methods	Options	return	Description

registerToServer
(options)
	_sUserId
	: 사용자 ID
	_fCallback : 실행 후 호출될 callback 함수입니다.
		{
		"result" : Boolean(true : 성공, false : 실패),
		"resultCode" : String(결과 코드),
		"resultMessage" : String(결과 메세지)
		}
			푸시서버에 사용자 등록 후 결과 값을 반환합니다. _fCallback의 파라미터로 반환됩니다.

Class

`bizMOB.App`
- bizMOB Native App 제어 플러그인

Description

Methods	Options	return	Description
exit (options)	_sType(Default : kill) : 어플리케이션 종료 유형입니다. kill : 어플리케이션 종료, logout : 어플리케이션 재시작.	Void	어플리케이션을 종료합니다.
callPlugIn (options)	Id: 커스텀 플러그인의 id Param : 커스텀 플러그인에서 사용할 파라미터	{플러그인 리턴값}	커스텀 플러그인을 호출합니다.

Thank You

135-010 서울시 강남구 학동로 158(논현동 127) 율암빌딩 2층
T.02-563-5325, F.02-568-9534