

Tizen Native Application 개발 가이드

이해하기 쉬운 70개의 예제로 타이젠 플랫폼을 완전분석한다.

Author: Jung, Dong-Geun (Denis.Jung)

Except as noted, this content - excluding the Code Examples - is licensed under Creative Commons Attribution 3.0 and all of the Code Examples contained herein are licensed under BSD-3-Clause. For details, see the Content License

목차

1. EFL 소개	6
2. 타이젠 개발환경 구축	13
3. 타이젠 개발환경 실행	24
4. 샘플 예제 실행	34
5. BasicUI 예제 만들기	45
6. Label 위젯 사용방법	66
7. Button 위젯 사용방법	74
8. Bg 위젯으로 배경 만들기	91
9. Conformant 컨테이너로 화면 리사이즈	102
10. Entry 위젯 사용방법	110
11. Check 위젯 사용방법	123
12. Radio 위젯 사용방법	134
13. Popup 사용방법	146
14. Slider 위젯 사용방법	166
15. List 위젯에 텍스트 Item 추가	175
16. GenList 위젯에 아이콘 표시	185
17. Gallery 복합 위젯 만들기	199
18. WebView 위젯으로 만드는 간단한 웹브라우저	211
19. Layout으로 구현하는 탭화면	222
20. Naviframe으로 구현하는 헤더 & 네비게이션바	231

21. Box 컨테이너로 위젯을 순차적으로 배치하기.....	242
22. Scroller로 Sub 페이지 만들기.....	253
23. 문자열 사용방법.....	266
24. 문자열 구조체 Eina_Strbuf	277
25. 배열 구조체 Eina_List	286
26. 타이머 사용방법.....	299
27. 날짜 & 시간	306
28. Calendar 예제	320
29. Mouse Touch 이벤트 구하기	334
30. 계산기 예제.....	343
31. 캔버스에 그라데이션 출력	359
32. 캔버스에 사각형 출력	370
33. 캔버스에 다각형 출력	375
34. 캔버스에 텍스트 출력	383
35. 캔버스에 이미지 출력	388
36. 커스텀 Button 제작.....	397
37. Animator 사용방법	414
38. Audio 재생	435
39. Video 재생	450
40. Audio 녹음	469
41. 카메라 캡쳐	489
42. 시스템 정보.....	512

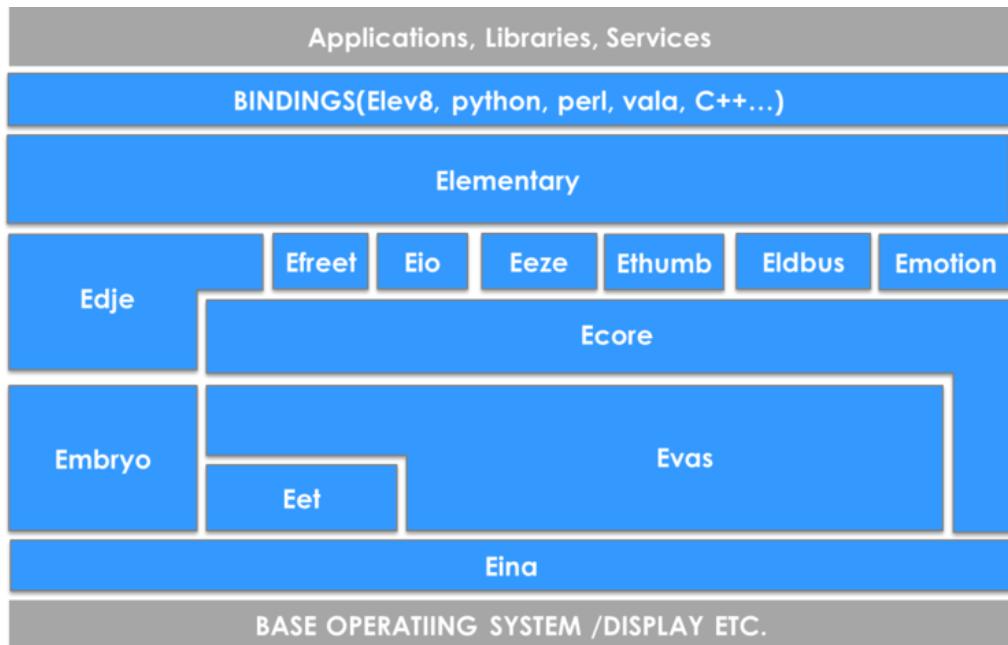
43. 시스템 환경설정 정보	526
44. 배터리 상태 정보	537
45. 진동 발생	547
46. LED 플래쉬 백라이트	560
47. 화면 회전 방향 이벤트	569
48. 하드웨어 키 이벤트 & 디버그 모드.....	581
49. 라이프 사이클 & 디버그 모드.....	590
50. Notify 사용방법	600
51. Acceleration 센서 사용방법	615
52. Gravity 센서 사용방법	630
53. Orientation 센서 사용방법	639
54. 자기장 센서 사용방법	651
55. Proximity 센서 사용방법.....	662
56. GPS 센서 사용방법	671
57. 구글맵 라이브러리	684
58. 텍스트 파일 읽고 쓰기	727
59. 파일 목록 구하기	740
60. Preference 사용방법	759
61. SQLite로 만드는 성적표 예제	770
62. AppControl로 외부앱 호출하기	790
63. 서비스 앱 제작하기	812
64. Alarm – 지정된 시간에 앱 실행하기	822

65. TTS (Text to Speech)	836
66. 다국어 지원	850
67. JSON 파싱	864
68. XML 파싱	874
69. Network 상태 확인	890
70. Http 통신	903
71. Wearable 소스 프로젝트 생성	915
72. 웨어러블 시스템 정보	925
73. Pressure 센서 사용방법	934

1. EFL 소개

타이젠 네이티브 API는 EFL을 메인으로 구성되어 있으며, EFL은 여러 가지 라이브러리가 합쳐진 복합 라이브러리입니다.

다음은 EFL 블록 다이어그램 입니다.



상위 라이브러리는 하위 라이브러리를 참조하고 있습니다. 예를 들어서, Elementary는 하위의 모든 라이브러리들에 의존하고 있으며 Ecore 같은 경우는 Evas, Eet, Eina를 참조하게 됩니다.

1) Eina

EFL의 가장 기본이 되는 Eina는 자료구조 라이브러리입니다. C++ 의 STL과 같이 배열, 리스트, 해시, 트리 그리고 공유 문자열(shared string)과 같은 다소 복잡한 로직을 사용자가 쉽고 빠르고 안전하게 구현할 수 있도록 유용한 기능들을 제공합니다. Eina 상단의 모든 EFL 라이브러리 뿐만 아니라, 어플리케이션에서도 이러한 데이터 구조 기능들을 이용하여 필요한 로직을 효과적으로 구현할 수 있습니다.

2) Eet

Eet는 데이터 인코딩 및 디코딩의 역할을 제공합니다. 임의의 자료구조나 이미지 데이터 등을 압축하여 파일로 저장하거나 네트워크를 통해 다른 머신으로 전송할 수 있으며 후에 다시 Eet를 이용하여 압축된 파일을 읽고 디코딩할 수 있습니다. Eet에서 사용되는 이러한 압축기법은 zip과 매우 유사하며 파일로 부터 임의의 위치의 데이터에 매우 빠르게 접근할 수 있습니다. 데이터 압축시 암호화하는 기능을 사용하면 안정적으로 데이터를 보관할 수 있습니다.

3) Evas

Evas는 캔버스의 역할을 담당합니다. 사용자는 Evas를 통해 윈도우 내에 이미지, 사각형, 선, 폴리곤 그리고 텍스트 등을 표현할 수 있습니다. 모든 출력물은 객체화가 되어있습니다. Evas에서는 이러한 객체들을 Evas_Object 타입으로 제공하며 사용자는 EFL 프로그래밍에서 모든 그래픽 객체들을 Evas_Object라는 인터페이스로 접근하고 이를 통해 화면 상에 특정 그래픽 객체들을 표현할 수 있습니다. 또한, Evas는 각

객체가 사용자 입력 이벤트에 적절히 반응하도록 그 인터페이스도 제공합니다..

Evas는 렌더링 방식에 있어서 리테인 모드(retain mode)를 채택하였으며, 내부적으로 장면그래프(scene-graph)를 통해 객체들을 관리하면서 화면상에 보여야 할 객체들을 적절히 최적화하여 알아서 렌더링 해줍니다. 따라서 앱 개발자는 복잡한 렌더링 매커니즘에서 자유로워질 수 있습니다.

기본적으로 Evas는 소프트웨어 렌더링 방식을 지원하지만, 플랫폼 환경에 따라 그래픽 H/W 가속 기능을 벡엔드(backend)로 지원합니다.

4) Ecore

Ecore는 사용자들의 편의를 위해 제공되는 시스템 기반 라이브러리이며 메인루프, 타이밍, 이벤트, 커넥션, IPC, 스레드, 윈도우 시스템 등과 관련된 기능들을 제공하고 있습니다. 각 시스템 API들의 복잡한 설정 및 사용 단계들을 내부적으로 처리해주고 보다 쉽고 단순화하여 제공되므로, 사용자들이 직접 시스템 기능들을 구현할 때 필요한 시간과 노력이 단축될 수 있게 도와줍니다.

EFL 어플리케이션은 Ecore에서 제공하는 메인루프를 기반으로 작동되는데, Ecore에서 제공되는 기능들이 Ecore 메인루프의 주 로직에 접합되어 처리되므로 필요 시에는 시스템 API를 이용하지 말고 반드시 Ecore에서 제공되는 기능들을 이용해야 합니다.

5) Edje

Edje는 복잡한 GUI 구성을 위한 기능들을 제공합니다. Edje는 EDC라는 스크립트 언어를 제공하며 사용자는 EDC 스크립트를 통해 프로그램 코드로부터 GUI 부분을 분리하여 프로그램을 작성할 수 있습니다. EDC는 edje_cc 컴파일러를 통해 EDJ 바이너리 형태로 변환되는데, 프로그램은 이러한 EDJ 파일을 런타임시에 읽어서 Evas_Object로 바인딩하여 GUI를 구축할 수 있습니다. Edje의 이러한 특성 때문에 어플리케이션은 재컴파일 하지 않고 GUI 디자인을 바꿀 수도 있습니다.

6) Embryo

Embryo는 일종의 바이트코드 가상 머신으로서 EDC 파일 내에서 구현될 수 있는 작은 프로그램을 위해 이용됩니다. 일반적으로 사용자들은 EDC 내에서 단순한 계산을 구현하거나 각 오브젝트 상태를 바꾸는 등의 간단한 기능을 구현하기 위해 C 언어 스타일의 Embryo 스크립트 언어를 추가로 이용할 수 있습니다. 이러한 Embryo 스크립트는 PAWN 컴파일러를 통해 PAWN 바이너리로 변환됩니다. 결국, PAWN 프로그램은 머신의 환경에 의존하지 않고 Embryo의 AMX(Abstract Machine eXecutive) 가상머신에 의해 해독되어 작동될 수 있으며 하나의 출력물을 가지고도 다른 시스템 환경에서도 동일한 동작을 보장할 수 있게 합니다.

7) Emotion

Emotion(이모션)은 비디오/오디오 플레이백(Playback) 라이브러리입니다. Emotion은 Gstreamer, Xine 혹은 VLC와 같은 다른 비디오 재생 플러그인 등을 이용하여 영상을 재생하고 이러한 영상 출력 결과물을 Evas Object에 연동하여 사용자에게 제공합니다. 이를 통해 사용자는 동영상 재생과 함께 GUI를 결합한 화면 구성을 쉽고 간단히 구현할 수 있습니다.

8) Elementary

Elementary는 위젯 툴킷 라이브러리이며, 버튼, 리스트, 레이블, 슬라이더와 같은 범용의 위젯들을 제공합니다. 뿐만 아니라 다양한 룩 앤 필(Look & Feel)을 지원하기 위한 동적 테마 변경, 다양한 스크린 해상도를 지원하기 위한 GUI 확장성(Scalability) 등을 제공합니다.

9) Efreet, Eio, Eeze, Ethumb, Edbus

Efreet, Eio, Eeze, Ethumb, E_Dbus는 부가적으로 추가된 라이브러리입니다.

Efreet는 아이콘, 데스크톱 파일, 메뉴 등과 관련하여 freedesktop.org의 표준에 맞춰 애플리케이션이 작동할 수 있도록 설계된 라이브러리입니다.

Eio는 비동기 입출력(I/O)을 위한 라이브러리이고, Ethumb은 frame 이미지를 더한 썸네일 이미지를 생성해주는 기능을 제공합니다.

Eeze는 udev를 통하여 하드웨어 장비를 조작하기 위해 쓰입니다. 예로 cd-rom의 디스크 삽입 여부, CPU 온도, 전원의 뱃데리와 같은 장비의 상태 및 정보를 얻고 조작하는데 사용됩니다. 마지막으로 Edbus는 메세지 버스 시스템인 dbus의 랩퍼(wrapper)로 볼 수 있으며, IPC를 수행하는데 있어서 dbus의 명세서(specification)를 구현하고 있습니다.

10) Language Bindings

EFL은 기본적으로 C 언어를 지원하고 있으며, 그 외에 Elev8(JavaScript), 파이썬, Perl, C++, Vala와 같은 랭귀지 바인딩 프로젝트를 진행하고 있습니다.

(출처 : EFL 한국 커뮤니티)

2. 타이젠 개발환경 구축

가. 자바 개발 도구 JDK(Java Development Kit) 다운로드

타이젠 개발환경은 모든 운영체제를 지원하기 위해서 자바를 사용합니다. 따라서 JDK(Java Development Kit)가 설치되어 있어야 합니다. JDK가 설치되어 있지 않으면 타이젠 SDK를 설치하는 도중에 오류메시지가 표시되고 설치가 중단됩니다. 아래 과정을 따라하면 됩니다.

1) 웹 브라우저를 실행하고

<http://www.oracle.com/technetwork/java/javase/downloads>로 접속합니다.

The screenshot shows the Oracle Java Technology Network website. The navigation bar includes links for Sign In/Register, Help, Country, Communities, I am a..., I want to..., Search, Products, Solutions, Downloads, Store, Support, Training, Partners, About, and OTN. The main content area is titled "Java SE Downloads" and features two download options: "Java Platform (JDK) 7u51" and "NetBeans 7.4". Each option has a "DOWNLOAD" button. To the right, there are sections for "Java SDKs and Tools" (including Java SE, Java EE and Glassfish, Java ME, JavaFX, Java Card, NetBeans IDE, and Java Mission Control) and "Java Resources" (including Java APIs, Technical Articles, Demos and Videos, Forums, Java Magazine, Java.net, Developer Training, Tutorials, and Java.com). A sidebar on the left lists categories like Java SE, Java EE, Java ME, Java SE Support, Java SE Advanced & Suite, Java Embedded, JavaFX, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, and Java Magazine.

2) 'Java DOWNLOAD' 버튼을 (또는 JDK DOWNLOAD) 누르고 화면이 바뀌면 'Accept License Agreement'에 체크합니다.

3) Java SE Development Kit xux를 다운로드 합니다. 여러가지 목록 중에서 자신이 사용하는 운영체제에 해당하는 버전을 선택하면 됩니다.
32비트 운영체제를 사용한다면 Window x86을 클릭하고, 64비트 운영체제를 사용한다면 Window x64를 클릭하면 됩니다.

Overview Downloads Documentation Community Technologies Training

Java SE Development Kit 7 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

Looking for JavaFX SDK?
JavaFX SDK is now included in JDK 7 for Windows, Mac OS X, and Linux x86/x64.

See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum

Java SE Development Kit 7u51

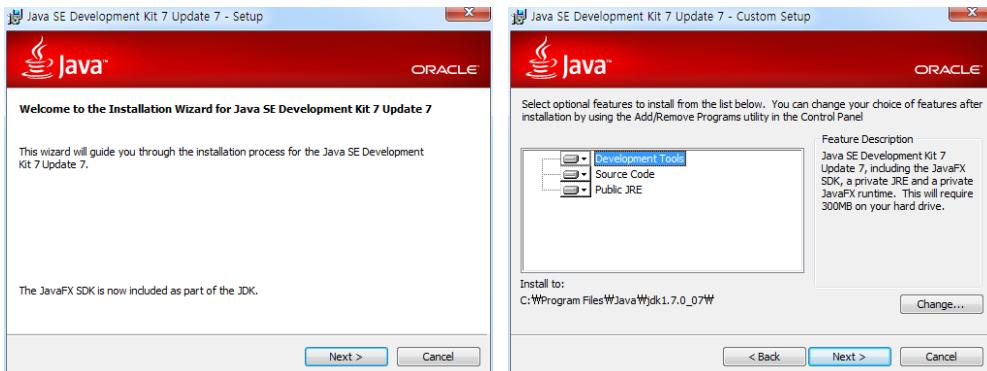
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	67.7 MB	jdk-7u51-linux-arm-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	67.68 MB	jdk-7u51-linux-arm-vfp-sflt.tar.gz
Linux x86	115.65 MB	jdk-7u51-linux-i586.rpm
Linux x86	132.98 MB	jdk-7u51-linux-i586.tar.gz
Linux x64	118.96 MB	jdk-7u51-linux-x64.rpm
Linux x64	131.8 MB	jdk-7u51-linux-x64.tar.gz
Mac OS X x64	179.49 MB	jdk-7u51-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.02 MB	jdk-7u51-solaris-i586.tar.Z
Solaris x86	95.13 MB	jdk-7u51-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.53 MB	jdk-7u51-solaris-x64.tar.Z
Solaris x64	16.28 MB	jdk-7u51-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	139.39 MB	jdk-7u51-solaris-sparc.tar.Z
Solaris SPARC	98.19 MB	jdk-7u51-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	jdk-7u51-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.33 MB	jdk-7u51-solaris-sparcv9.tar.gz
Windows x86	123.64 MB	jdk-7u51-windows-i586.exe
Windows x64	125.46 MB	jdk-7u51-windows-x64.exe

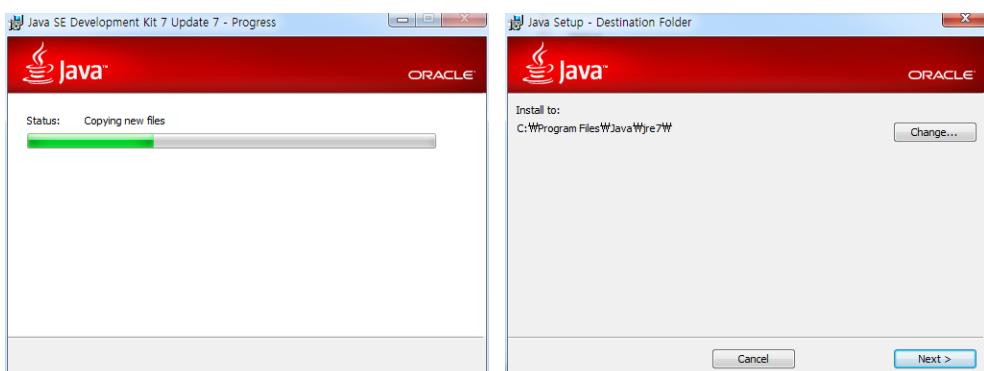
나. 자바 개발 도구 JDK(Java Development Kit) 설치

1) 다운로드한 실행파일을 더블클릭하면 설치화면이 나타납니다. Next 버튼을 클릭합니다.



2) 설치할 항목을 선택하는 화면이 나타나면 기본 설정 그대로 놔두고 Next 버튼을 클릭합니다.

3) JDK 설치가 시작되면 끝날때까지 기다립니다. JRE 설치경로를 물어보는 화면이 나타나면 Next 버튼을 클릭합니다. 설치경로를 기본 폴더 (ex: C:/Program Files/Java/jre7/) 이외에 다른 폴더로 지정하고 싶다면 Change 버튼을 클릭하고 변경하면 됩니다.



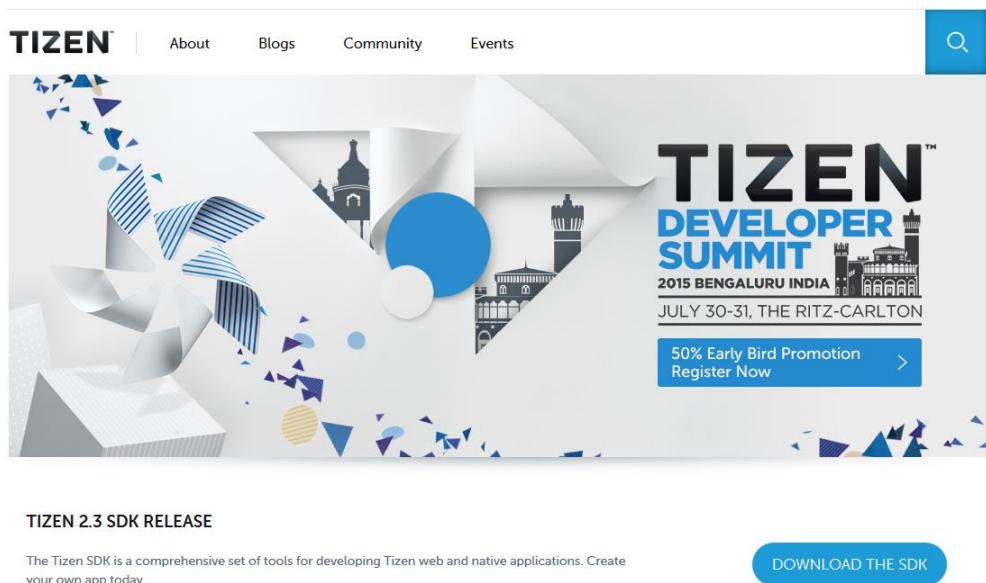
4) JRE 설치가 시작되면 끝날 때까지 기다립니다. 설치가 끝났다는 화면이 나타나면 Close 버튼을 클릭하고 설치를 종료합니다.



다. 타이젠 SDK 다운받기

타이젠 개발환경을 구축하는 방법을 알아보겠습니다. 우선 SDK 설치 파일을 다운로드 하겠습니다.

<https://www.tizen.org/>로 접속 합니다.



첫 화면에서 'DOWNLOAD THE SDK' 버튼을 클릭하면 SDK 다운로드 페이지로 이동합니다.

윈도우 32비트 운영체제를 사용한다면 'Install Manager' 목록 중에서 'Windows® XP/7 32bits'를 클릭해서 다운받고, 64비트 운영체제를 사용한다면 'Windows® XP/7 64bits'를 다운받으면 됩니다. 그 외에 우분투 32bits/64bits, 맥OS를 지원합니다.

Tizen 2.3 Rev3 SDK

Install Manager

Platform	Install Manager	File Size	MD5 Checksum	Updated Date
Ubuntu® 32bits	tizen-sdk_2.3.63_ubuntu-32.bin Alternative Locations: Global Brazil China India	5.5M	b1efdf3b4393cace59fbaa0503708241	Feb 13, 2015
Ubuntu® 64bits	tizen-sdk_2.3.63_ubuntu-64.bin Alternative Locations: Global Brazil China India	5.7M	742274e6700b91d2e121e05397a5665e	Feb 13, 2015
Windows® 7 32bits	tizen-sdk_2.3.63_windows-32.exe Alternative Locations: Global Brazil China India	6.0M	e3f3c4f1cb769c4f0c5f66d87d42d05e	Feb 13, 2015
Windows® 7 64bits	tizen-sdk_2.3.63_windows-64.exe Alternative Locations: Global Brazil China India	6.0M	f7867b2e7dba47707fd1ed521c4c0bda	Feb 13, 2015

Install Manager 파일을 다운받았으면 아래로 스크롤해서 'SDK Image'도 동일한 항목을 다운 받으면 됩니다. 'Install Manager'만 다운받으면 설치하는 중에 자동으로 필요한 파일을 웹에서 다운로드 합니다. 설치하는 동안 오류가 발생하지 않기 위해서 이미지 파일을 미리 다운로드 하는 것이 좋습니다.

SDK Image

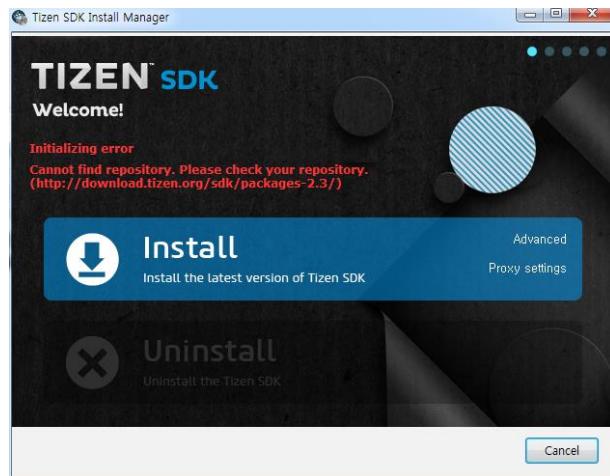
Platform	SDK Image	File Size	MD5 Checksum	Updated Date
Ubuntu® 32bits	tizen-sdk-image-TizenSDK_2.3.0_Rev3-ubuntu32.zip Alternative Locations: Global Brazil China India	2.1G	b448ebb652cfb1fc7737e3d30a71ca39	July 1, 2015
Ubuntu® 64bits	tizen-sdk-image-TizenSDK_2.3.0_Rev3-ubuntu64.zip Alternative Locations: Global Brazil China India	2.1G	ae0f7fa8a25aa7bd563f15e1718e11ac	July 1, 2015
Windows® 7 32bits	tizen-sdk-image-TizenSDK_2.3.0_Rev3-windows32.zip Alternative Locations: Global Brazil China India	2.5G	360767f03103ef05ec1ac17f190d7f3e	July 1, 2015
Windows® 7 64bits	tizen-sdk-image-TizenSDK_2.3.0_Rev3-windows64.zip Alternative Locations: Global Brazil China India	2.5G	0f16e4293215546dfa0072934abe1917	July 1, 2015

라. 타이젠 SDK 설치하기

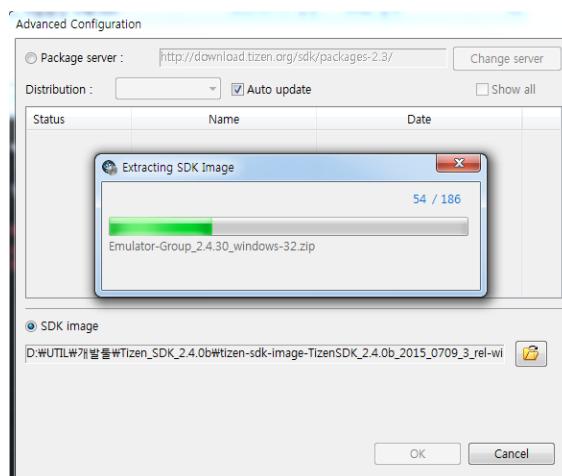
타이젠 SDK를 설치하는 방법을 알아보겠습니다. SDK 2.4.0b를 기준으로 설명하겠습니다. 다른 버전이라도 크게 다르지 않습니다. 그리고 타이젠 SDK는 설치가 간편해서 초보자도 설명서 없이 쉽게 설치할 수 있습니다.

다운로드가 완료되면 설치파일(tizen_sdk_2.x.xx_window-32.exe) 을 더블클릭해서 실행합니다. 만약 경고창이 나타나면 무시하고 'Yes' 버튼을 클릭하면 됩니다.

설치 화면이 나타나면 이미지 파일을 지정해 주어야 합니다. 우측에 'Advanced' 버튼을 클릭합니다.

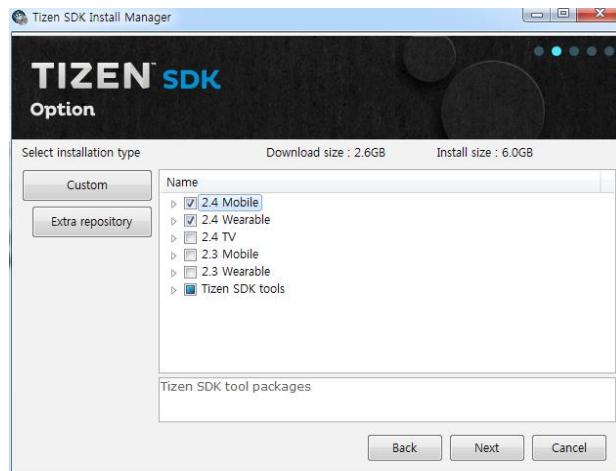


'SDK Image'에 체크하고 우측에 파일 열기 버튼을 클릭하면 파일 열기 팝업창이 나타납니다. 타이젠 개발자 사이트에서 다운받은 이미지 파일(tizen-sdk-image-TizenSDK_2.4.x_xxxx_xxxx_x_rel-windows-32.zip) 을 선택합니다. 압축해제가 진행됩니다. 압축해제가 완료될 때까지 기다립니다.

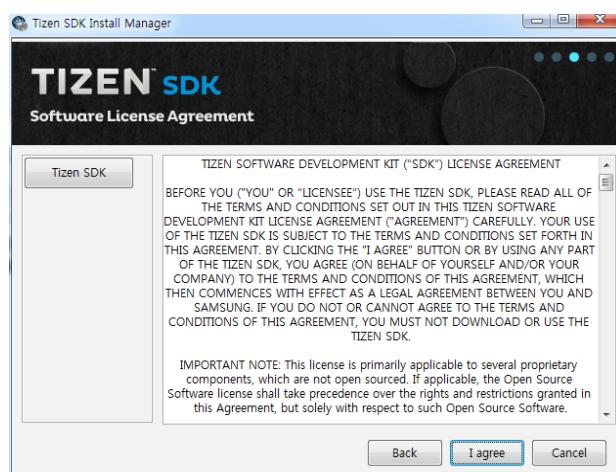


압축해제가 완료되면 OK 버튼을 클릭하고 빠져나옵니다. 메인화면에서 Install 버튼을 클릭하고 다음 페이지로 넘어갑니다.

'Select Install Type' 화면이 나타나면 2.4 Mobile과 2.4 Wearable에 체크하고 Next 버튼을 클릭합니다.



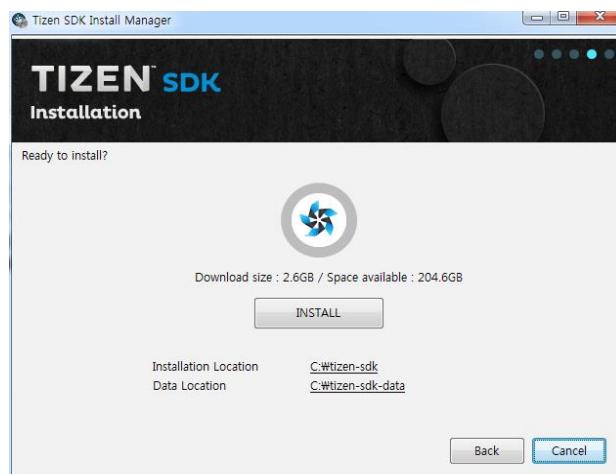
'Software License Agreement' 화면이 나타나면 'I agree' 버튼을 클릭합니다.



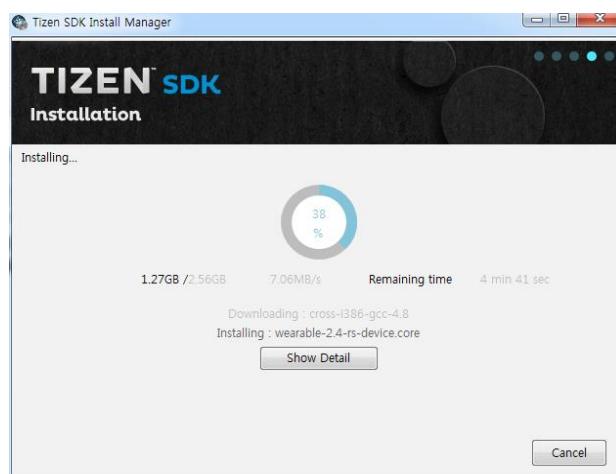
'Ready to install' 화면에서는 SDK가 설치될 경로와 SDK 데이터 폴더 경로를 지정합니다. SDK 기본 경로는 C:\Tizen-sdk 입니다. 변경하고 싶으면 경로 주소를 클릭하고 폴더를 지정하면 됩니다.

데이터 폴더의 기본 경로는 C:\Tizen-sdk\data 입니다. 복잡한 폴더 구조를 싫어한다면 C:\Tizen-sdk\data 처럼 SDK 폴더 내부로 수정하기 바랍니다.

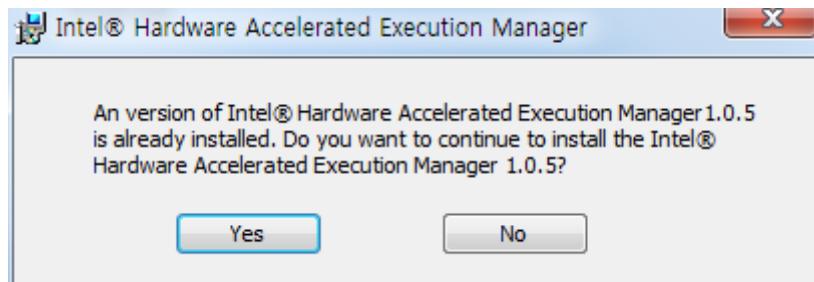
Install 버튼을 클릭하면 설치가 시작됩니다.



설치가 시작되면 완료될 때까지 기다립니다.



만약 설치하는 도중에 하드웨어 가속 드라이버를 업데이트 해야한다는 팝업창이 나타나면 Yes 버튼을 누르고 설치를 진행하면 됩니다.



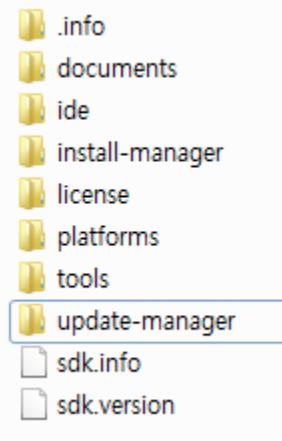
설치가 완료되면 Close 버튼을 클릭하고 종료합니다.



3. 타이젠 개발환경 실행

가. 설치된 SDK 둘러보기

설치된 SDK에 어떤 자료가 포함되어 있는지 살펴보도록 하겠습니다. SDK 폴더를 열어보면 아래와 같은 폴더 목록이 나타납니다. 기본 설정 그대로 설치했다면 C:/tizen-sdk가 설치폴더 입니다.



이 중에서 중요한 폴더 몇가지만 살펴보겠습니다. #documents 폴더에는 타이젠 API와 개발에 관련된 설명서가 PDF 파일 형식으로 저장되어 있습니다.

#platforms 폴더에는 버츄얼 디바이스(에뮬레이터)와 타이젠 API를 사용하는 여러가지 예제가 저장되어 있습니다.

네이티브 앱 예제는 `#platforms#tizen-x.x#mobile#samples#Template#Native` 폴더에 저장되어 있고,

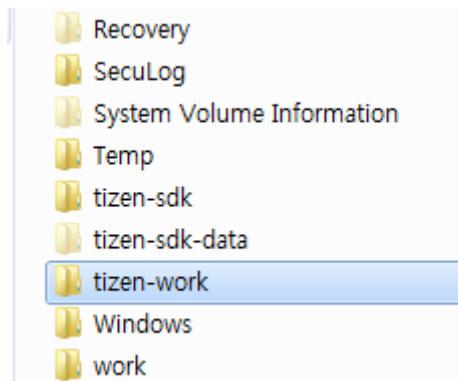
웹 앱 예제는 `#tizen-sdk#platforms#tizen-x.x#mobile#samples#Template#Web` 폴더에 저장되어 있습니다. 소스 프로젝트 전체가 들어있기 때문에 IDE에서 로딩하여 빌드하고 바로 실행시켜서 결과를 확인할 수 있습니다. 개발자들이 가장 많이 봐야 할 자료가 바로 여기에 있습니다.

`#Wide` 폴더에는 타이젠 어플리케이션을 제작할 수 있는 개발 환경입니다. 새로운 프로젝트를 생성하여 소스코드를 입력하고, UI 에디터에서 화면에 위젯을 생성 및 속성을 지정하고, 빌드하며, 에뮬레이터 및 타겟으로 실행시킬 수 있는 IDE가 저장되어 있습니다. 앱 개발자가 이 폴더를 들여다 볼 필요는 없습니다.

`#tools` 폴더에는 인증서를 생성하는 도구와 리눅스에서 소스 프로젝트를 빌드하고 실행하는 도구들이 들어있습니다.

바. IDE-개발환경 툴 실행하기

실제 개발환경을 실행시켜 보도록 하겠습니다. IDE를 실행하기 전에 먼저 작업 폴더를 생성하겠습니다. C: 드라이브 루트 폴더에 새로운 폴더를 생성하고 이름을 tizen-work 라고 지정하겠습니다. 본인이 원하는 경로에 원하는 이름으로 지정해도 상관없습니다.



이제 설치된 SDK에서 타이젠 IDE를 실행해 보겠습니다. 자세한 경로는 윈도우 시작메뉴에서 [모든 프로그램 > Tizen SDK > Tizen IDE] 이 경로를 따라가면 됩니다.

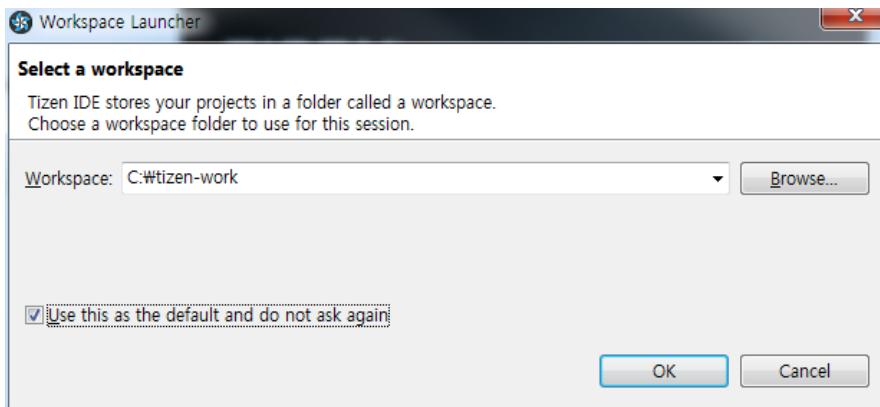
[Tip!]

타이젠에서는 IDE(개발도구)로 이클립스를 사용합니다. 자바 & 안드로이드와 개발툴이 동일한 것입니다.

타이젠 네이티브 앱에서 사용하는 언어는 자바가 아니라 C언어이기 때문에 Visual Studio를 사용해도 상관없으나 이렇게 되면 개발자들이 각자 라이센스 비용을 지불하고 Visual Studio를 구입해야 하는 불편함이 있습니다. 그래서 개발자들이 무료로 개발 할 수 있도록 배려하는 차원에서 이클립스를 기본 탑재한 것입니다.

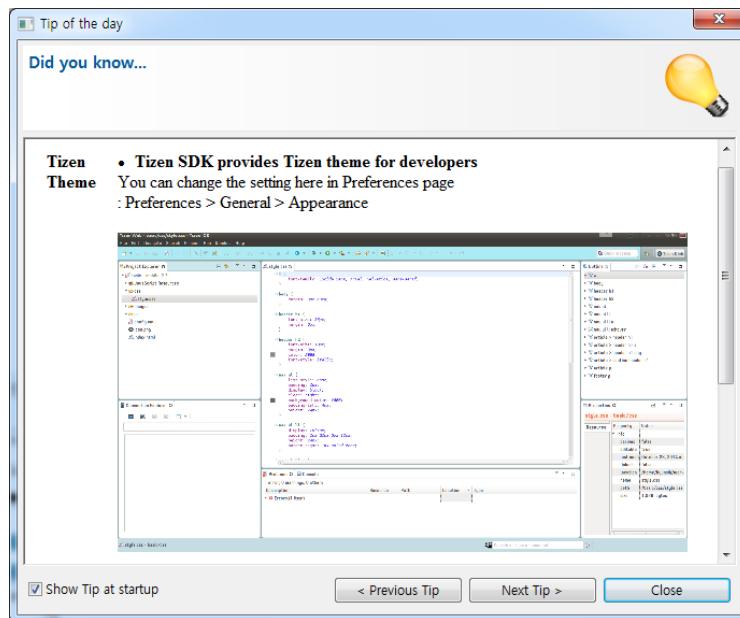
Visual Studio 또는 다른 개발환경에 익숙한 개발자들은 이클립스가 낫설겠지만 조금만 지나면 금방 친숙해 질것입니다.

처음으로 IDE를 실행할 때는 아래와 같은 팝업창이 나타납니다. 소스 프로젝트가 저장될 작업 폴더를 지정하는 화면입니다.

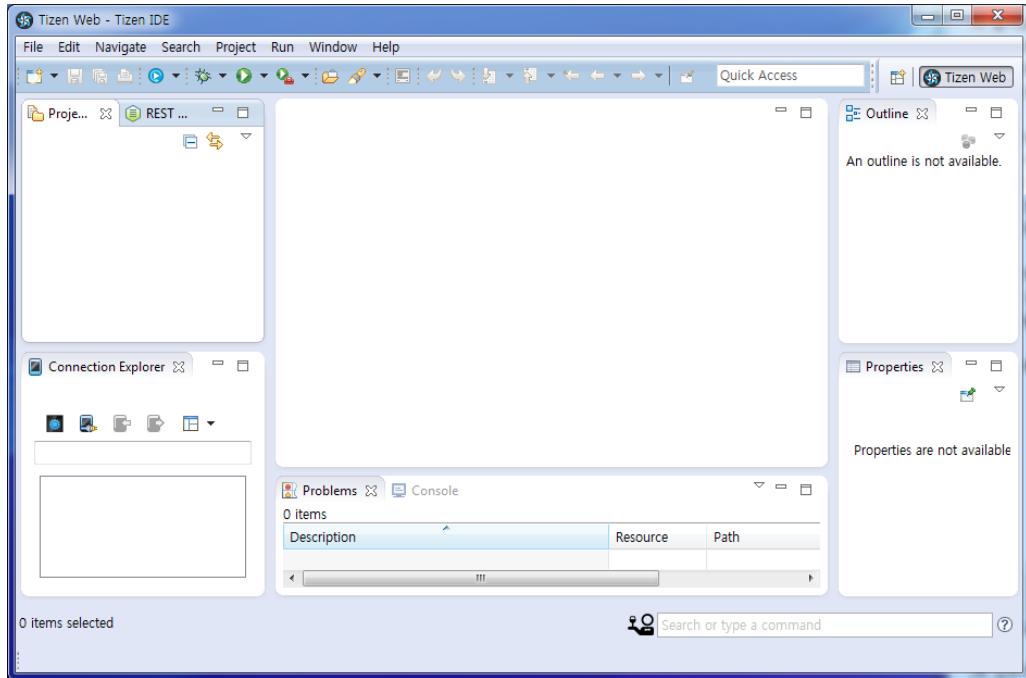


'Browse' 버튼을 누르고 워크스페이스로 사용할 폴더를 지정합니다. 필자의 경우는 C:\tizen-work를 작업 폴더로 사용합니다. 폴더를 지정했으면 'Use this as the default and do not ask again'에 체크를 하고 OK 버튼을 클릭해서 넘어가면 됩니다. 체크하지 않으면 매번 IDE를 실행할 때마다 이 창이 뜨기 때문에 체크하는 것이 좋습니다.

IDE가 실행되면 아래와 같은 화면이 나타납니다. 이것은 Tip을 소개하는 화면입니다. 우측상단에 'x' 기호를 클릭해서 닫으면 됩니다.



안내 페이지가 사라지면 아래와 같은 화면이 나타납니다. 자바나 안드로이드 개발을 해보신 분들에게는 친숙한 화면일 것입니다.



각 영역에 대해서 간략히 설명해 보겠습니다. 위쪽에는 메인메뉴와 툴바가 위치하고 있습니다. 저는 개인적으로 단축키 또는 메뉴를 주로 사용하다 보니 툴바는 거의 사용하지 않습니다.

툴바의 오른쪽에는 'Tizen Web'이라는 사각형이 있는데 이것은 웹앱과 네이티브앱, 그리고 디버그 모드와 일반 실행모드를 서로 전환할 때 사용합니다. 어플리케이션을 디버그 모드로 실행하게 되면 여기에 버튼이 하나 추가됩니다. 토글형식인 2개의 버튼으로 디버그 모드와 일반 실행모드를 왔다갔다 할 수 있습니다.

툴바의 좌측아래에는 'Project Explorer' 영역이 있습니다. 이곳에는 작업 폴더에 저장되어 있는 모든 프로젝트와 그 하위에 있는 파일들(소스 파일 & 리소스 파일)의 목록이 나타납니다. 이곳에서 소스파일이나 UI 화면 정보 파일을 선택하면 화면 중앙에 내용이 표시됩니다. 단축메뉴에서 프로젝트의 속성을 변경하거나 실행할 수도 있습니다.

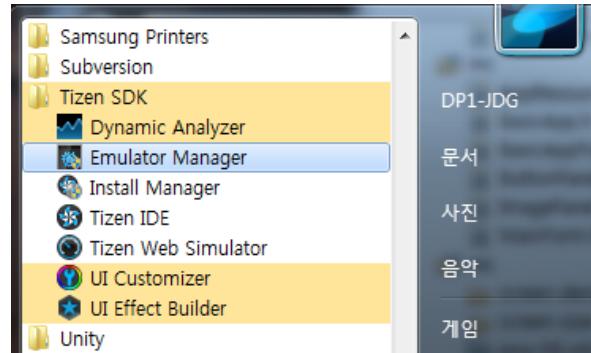
'Connection Explorer' 영역에는 개발한 앱을 설치할 수 있는 장비 목록이 나타납니다. 에뮬레이터를 실행하면 에뮬레이터가 표시되고 케이블로 폰을 연결하면 폰이 추가됩니다.

화면 하단 중앙에는 여러개의 탭화면이 보입니다. 중요한 것 몇 가지만 소개하겠습니다. 'Problems'는 프로젝트를 빌드할 때 발생하는 로그 메시지와 경고 또는 에러 메시지를 표시합니다. 빌드 오류가 발생했을 때 여기에 표시되는 메시지를 보고 수정할 수 있습니다.

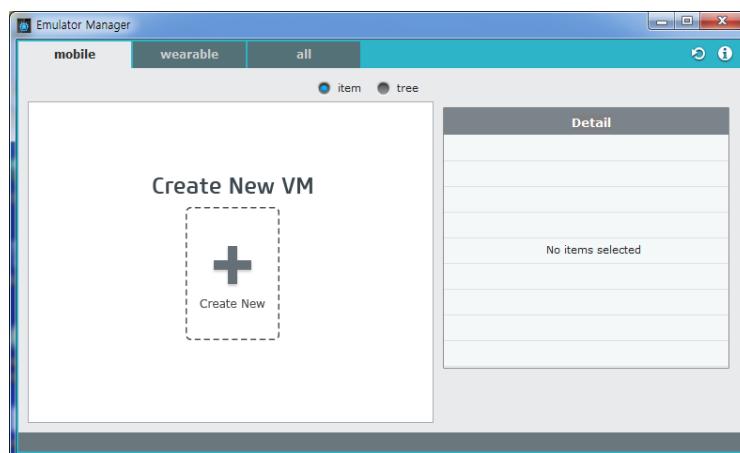
'Log'에는 어플리케이션을 실행하는 도중에 발생하는 로그메시지가 표시됩니다. AppLog() 또는 AppLogDebugTag()라는 API 함수를 소스코드에 사용해서 실행 도중에 진행상황을 확인할 수 있습니다.

사. 에뮬레이터 실행하기

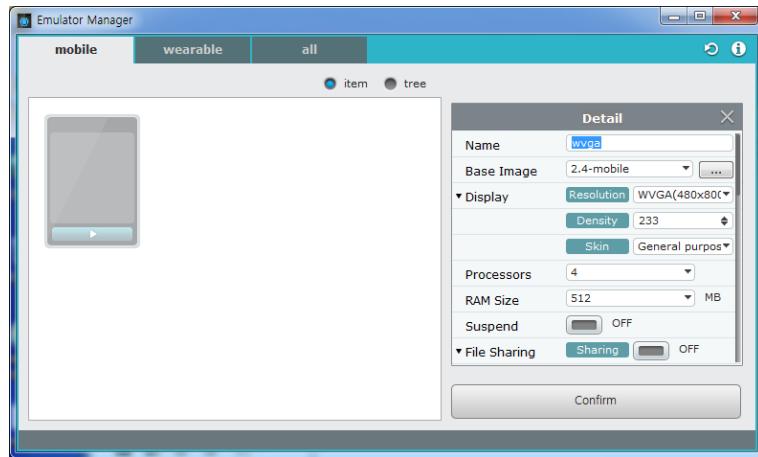
에뮬레이터를 생성하고 실행하는 방법을 알아보겠습니다. [원도우 시작버튼 > 모든 프로그램 > Tizen SDK > Emulator Manager]를 선택합니다. 만약 경고창이 나타나면 무시하고 ‘예’ 버튼을 클릭하면 됩니다.



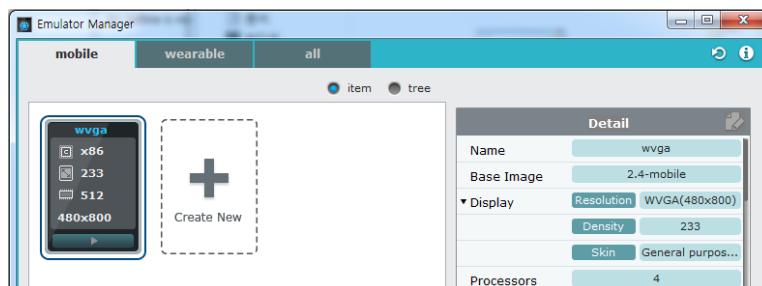
최초로 에뮬레이터 매니저를 실행했을 때는 에뮬레이터를 생성해야 합니다. 왼쪽 Create New VM라는 글자 아래에 + 기호를 클릭합니다.



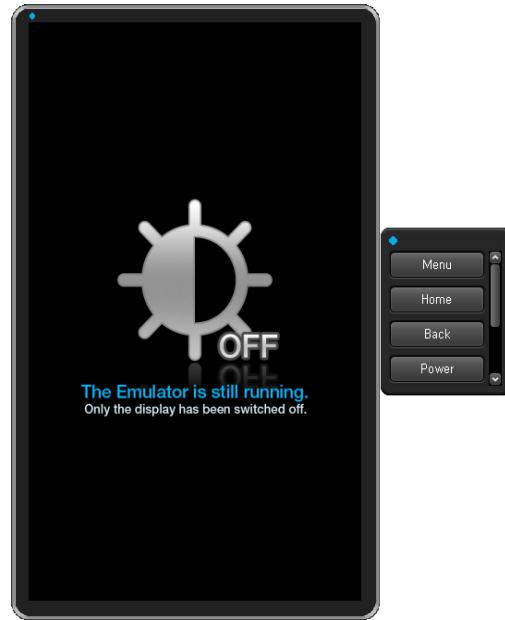
오른쪽에 에뮬레이터의 옵션을 지정하는 화면이 나타납니다. Name 속성에 wvga라고 지정하고 나머지는 디폴트 속성을 그대로 놔두어도 상관없습니다. Confirm 버튼을 클릭하면 에뮬레이터가 생성됩니다.



화면 왼쪽에 wvga라는 새로운 에뮬레이터가 생성되었습니다. 새로 생성된 에뮬레이터 아이콘의 아래쪽에 있는 화살표 버튼을 클릭하면 에뮬레이터가 실행됩니다. 만약 에뮬레이터 옵션을 수정하고 싶다면 Reset 버튼을 클릭하면 됩니다.



만약 'Windows 보안 경고' 팝업창이 나타나면 '차단 해제' 버튼을 누르고 계속 진행하면 됩니다. 에뮬레이터 오른쪽에는 작은 사각형 팝업창이 붙어있는데 여기에는 하드웨어 키들이 있습니다. 위에서부터 Menu, Home, Back, Power, Volume+ Volume- 순입니다. 안드로이드에서 지원하는 하드웨어 키와 거의 유사한 기능이라고 생각하면 됩니다.



슬립모드로 전환되었을 때는 Home 버튼을 클릭하면 화면이 다시 켜집니다. 이때 화면을 드래그해서 잠금을 해제합니다.



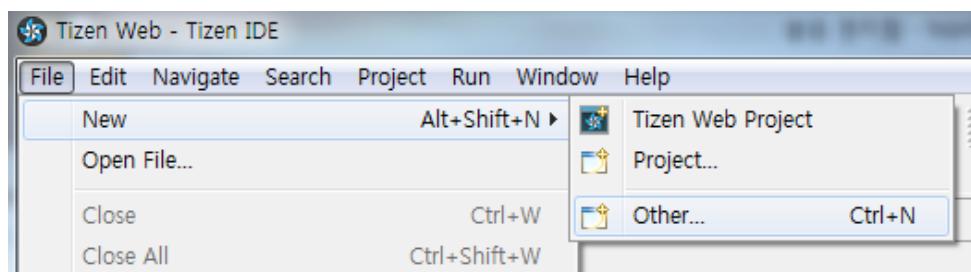
4. 샘플 예제 실행

SDK에는 가이드 문서도 있지만 역시나 가장 좋은 교재는 샘플 예제입니다. 여러분들이 가장 많이 참고해야 할 자료가 바로 샘플 예제들인 것입니다. 샘플 예제를 IDE에 추가하고 에뮬레이터로 실행하는 방법을 알아보겠습니다. 그리고 중요한 샘플 예제 몇 가지를 살펴보도록 하겠습니다.

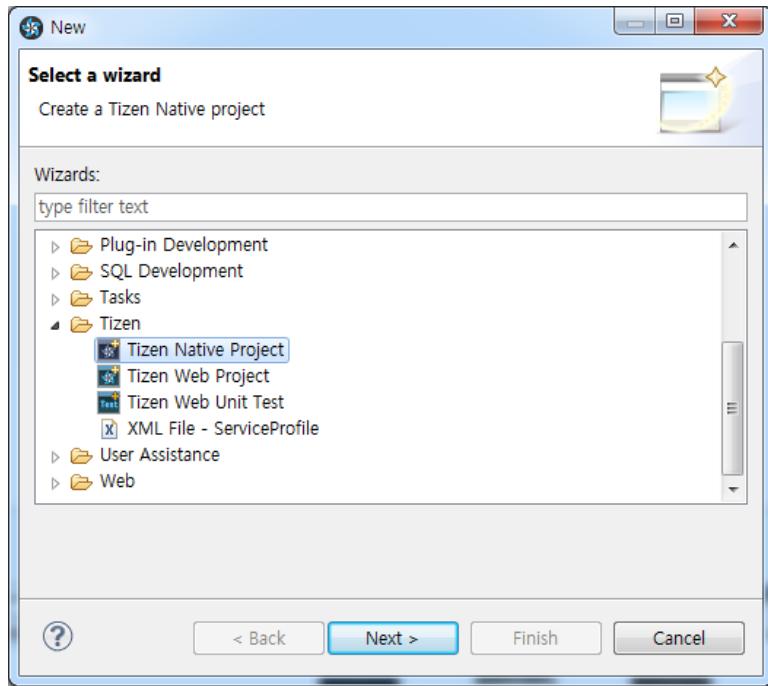
가. Sample 예제를 IDE에 추가하기

이제 타이젠 어플리케이션을 개발할 준비가 끝났습니다. 샘플 예제를 하나 실행해 보겠습니다.

메인메뉴에서 [File > New > Other...]를 선택합니다.

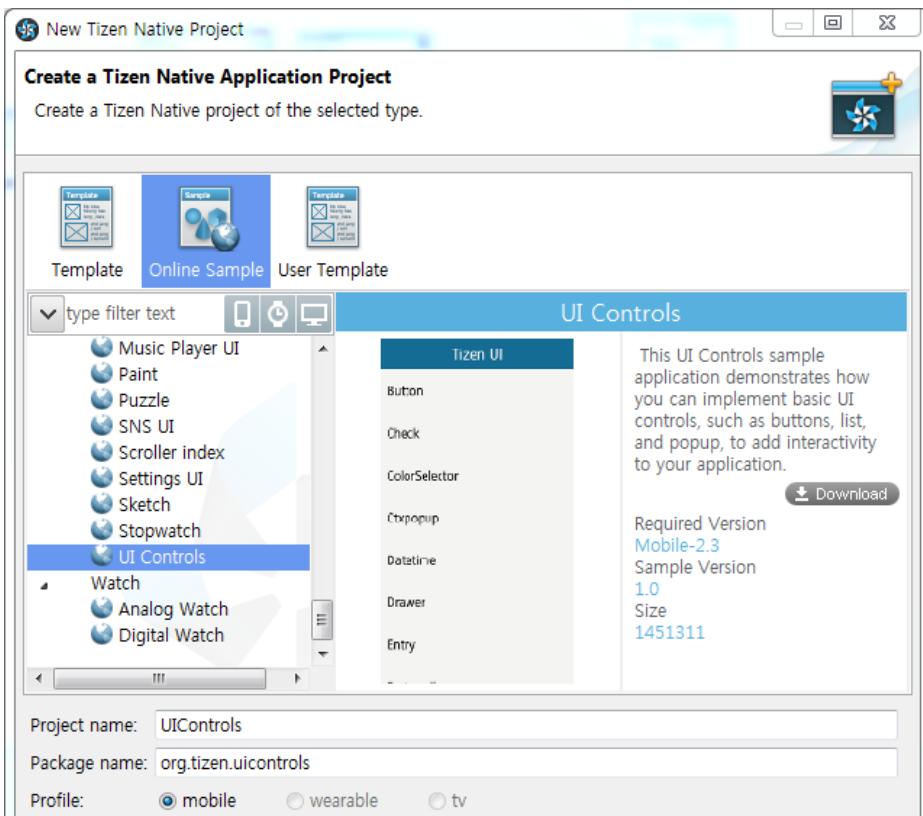


New 팝업창이 나타나면 트리목록이 보입니다. Tizen 폴더를 열고 항목들 중에서 'Tizen Native Project' 를 선택하고 Next 버튼을 클릭합니다.

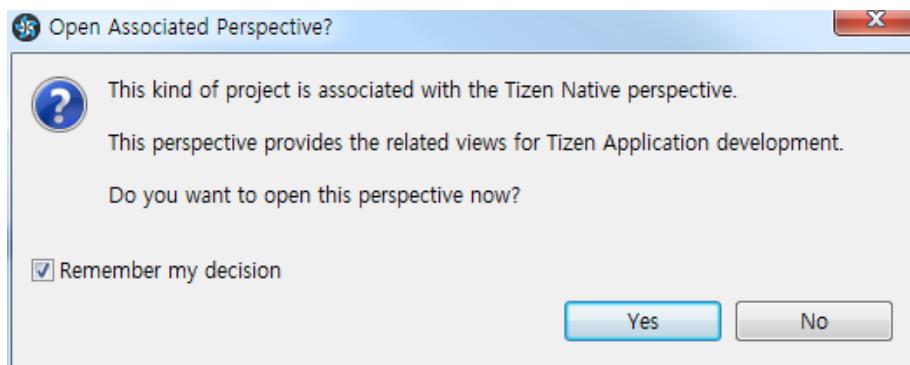


'Create a Tizen Native Application Project' 팝업창이 나타나면 Template 버튼 오른쪽에 있는 Online Sample 버튼을 클릭합니다. 네이티브 앱 예제 목록이 나타납니다. 왼쪽 목록에서 [UI > UI Controls]를 선택하면 오른쪽에 화면 캡쳐 이미지가 나타납니다.

프로젝트 이름을 지정해 주어야 합니다. 이건 이미 만들어진 예제를 불러오는 것이라서 원본 예제와 동일하게 지정해도 상관없습니다. 'Project name' 항목에 UIControls이라고 입력하고 Finish 버튼을 클릭합니다.



Perspective를 오픈 할 것인지를 묻는 팝업창이 나타나면 'Remember my decision'에 체크하고 Yes 버튼을 클릭합니다.

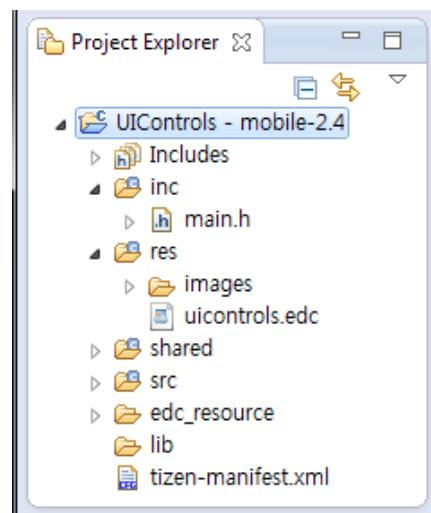


팝업창이 닫히고 해당 샘플 예제가 'Project Explorer'에 추가됩니다.

이 과정은 단순히 SDK의 /platforms 폴더에 있는 파일을 불러온 것이 아니라 자신의 작업 폴더에 복사한 후에 복사된 파일을 불러온 것입니다. 자신이 지정한 작업 폴더를(C:/tizen-work) 열어보면 UIControls 폴더가 복사된 것을 확인할 수 있습니다.

UiControls 는 타이젠에서 지원되는 모든 종류의 Widget 들과 Container의 사용방법을 보여주는 예제입니다.

'Project Explorer'에 추가된 BasicApp 왼쪽에 '+' 기호를 클릭하면 트리목록이 펼쳐지고 몇가지 폴더가 나타납니다. 각 폴더별 파일의 종류를 살펴보겠습니다.



/src 폴더에는 소스파일(.c)들이 저장되어 있습니다. /inc 폴더에는 헤더파일(.h)들이 저장되어 있습니다. 기본 프로젝트를 생성하면 자동으로 메인 화면이 생성됩니다.

새로운 화면이 추가될 때 메인 화면용 소스파일에서 작업해도 되지만 새로운 소스파일을 생성하는 편이 관리하기 쉽고 다른 프로젝트에서 재활용하기에도 쉽습니다.

라이브러리를 제작하는 경우에는 새로운 헤더파일에서 작업하는 것이 좋습니다.

앱 아이콘 이미지는 /shared/res 폴더에 저장되어 있습니다.
스마트폰에서 아이콘 목록 중 특정 아이콘을 선택하면 해당 어플리케이션이 실행되는데 이때 사용되는 이미지가 앱 아이콘 이미지입니다.

tizen-manifest.xml은 앱에 관한 각종 정보(어플리케이션 ID, SDK 버전, 어플리케이션의 버전, Privilege(사용자 지정 권한), 앱 아이콘 이미지, 앱 아이콘 Label 텍스트 등)를 지정하는 파일입니다.

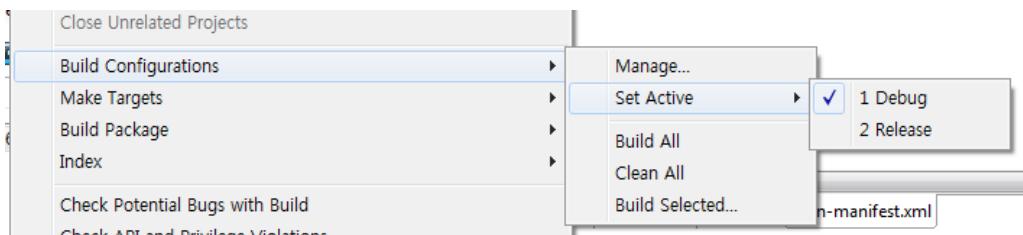
나. Sample 예제 빌드하기

소스 프로젝트를 에뮬레이터로 실행시키려면 먼저 실행파일을 빌드해야 합니다. 타이젠 네이티브 앱 개발에는 C 언어를 사용하기 때문에 자바나 안드로이드 처럼 자동으로 빌드되지는 않습니다. 또한 어떠한 모드로 빌드할 것인지를 선택해야 합니다.

빌드 모드는 디버그 (Debug)와 릴리즈(Release) 이렇게 2가지 모드가 있습니다. 디버그 모드에서는 실행 중에 로그 메시지로 원하는 정보를 Log 창에 표시하는 것이 가능하고 디버깅도 가능합니다.

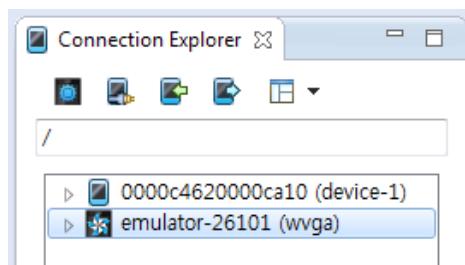
릴리즈 모드를 사용하면 로그 메시지를 확인할수는 없고 실행만 가능합니다. 개발은 디버그 모드로 했다가 최종적으로 셀러 사이트에 어플리케이션을 등록 할때는 릴리즈 모드로 빌드한 패키지를 사용해야 합니다.

디풀트 설정은 디버그 모드입니다. 빌드 모드를 변경하려면 'Project Explorer'에서 소스 프로젝트를 마우스 오른쪽 버튼으로 클릭하고, [Build Configurations > Set Active]에서 원하는 항목을 선택하면 됩니다.

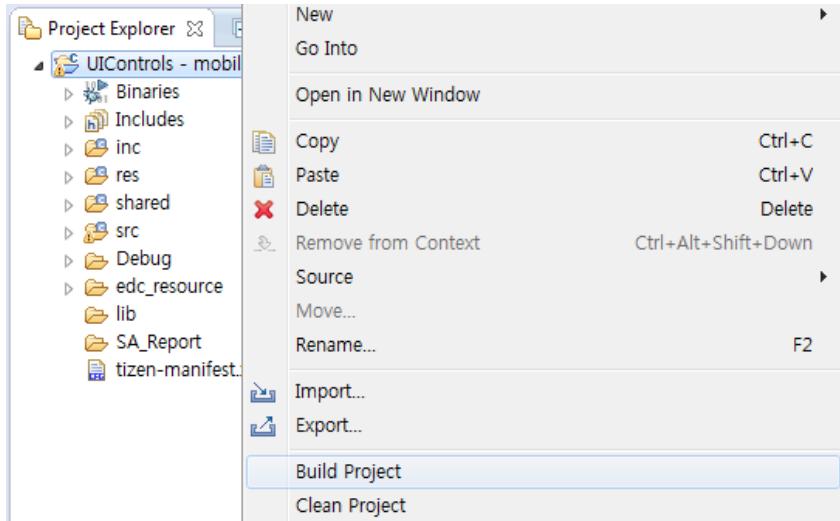


이 클립스 좌측 하단에는 Connection Explorer 창이 있습니다. 좀전에 생성한 에뮬레이터가 등록되어 있습니다. 만약 단말을 USB 케이블로 연결하면 추가 장비가 보입니다.

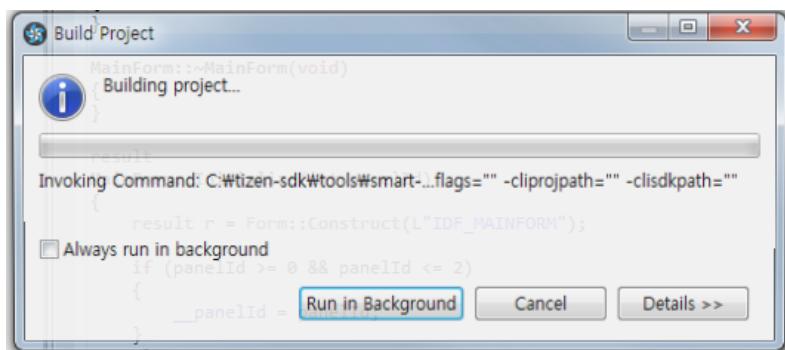
에뮬레이터에서 실행 할 때는 Connection Explorer 창에서 에뮬레이터를 선택한 상태로 빌드하고, 단말에서 테스트 할 때는 단말을 선택한 상태로 빌드하면 됩니다.



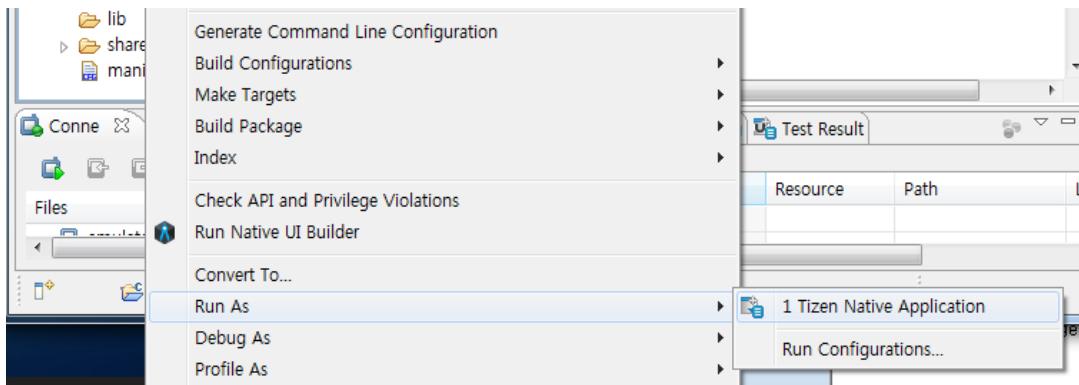
빌드 모드를 지정했으면 이제 빌드할 차례입니다. 프로젝트 단축메뉴에서 [Build Project]를 선택하면 빌드가 시작됩니다.



팝업창이 나타나서 빌드 진행율을 표시합니다. 이 창이 자동으로 사라질 때까지 기다리면 됩니다. 팝업창이 사라지고 IDE 아래쪽에 Problems 탭에 오류가 표시되지 않으면 성공적으로 빌드된 것입니다. 만약 오류가 표시되면 문제가 발생한 부분을 수정하고 다시 빌드하면 됩니다.

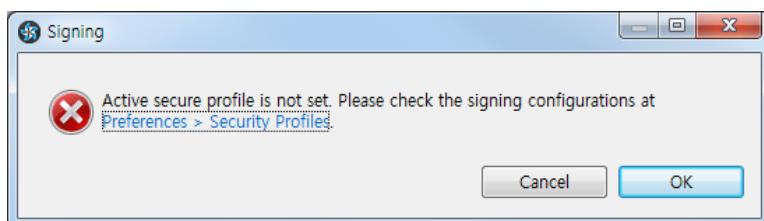


빌드가 완료되었으므로 이제 실행해 보도록 하겠습니다. 소스 프로젝트의 단축메뉴에서 [Run As > 1 Tizen Native Application]을 선택합니다.

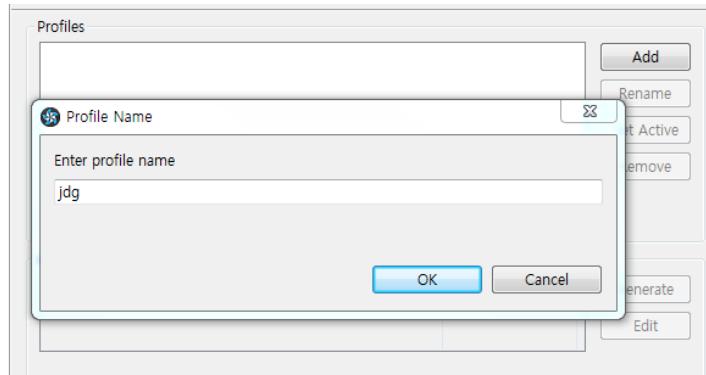


다. 인증서 생성하기

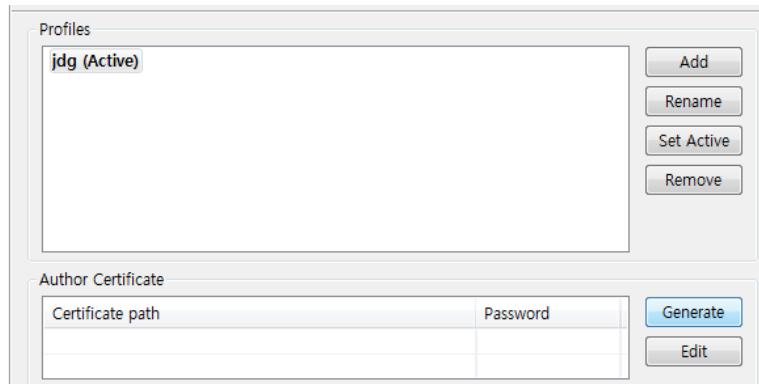
이때 Secure Profile이 존재하지 않는다는 경고창이 나타날수 있습니다.
타이젠 SDK 2.1 부터는 에뮬레이터 또는 단말에 앱을 설치하려면
인증서를 생성해야 합니다. 이제부터 인증서를 생성하는 방법을
알아보겠습니다.



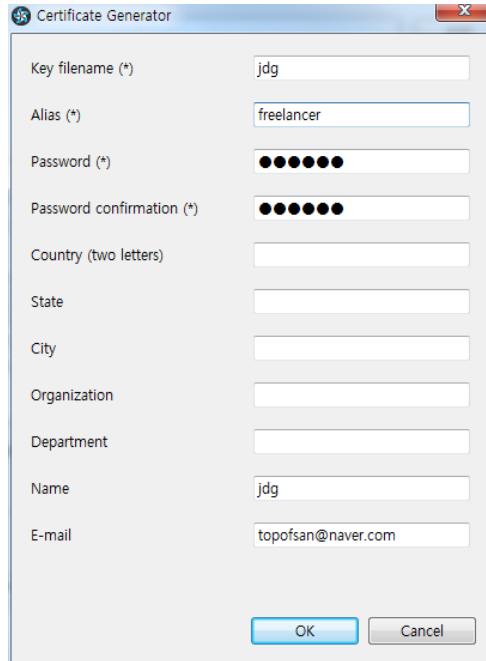
팝업창에서 Preferences > Security Profiles 링크를 클릭하면 새로운
팝업창이 나타납니다. 오른쪽에 있는 Add 버튼을 클릭하고 Profile Name
이라는 팝업창이 나타나면 프로파일 이름을 입력합니다. 보통
회사명이나 자신의 이름을 영문으로 입력하면 됩니다. OK 버튼을 누르면
팝업창이 닫힙니다.



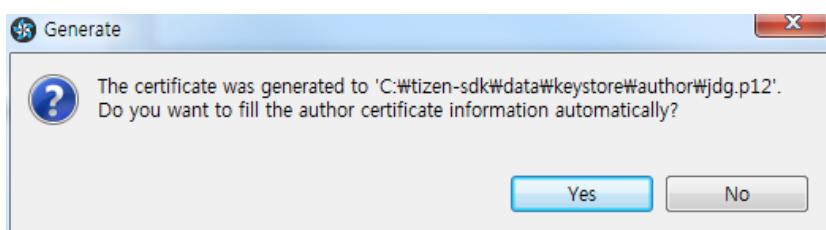
Profiles에 새로운 항목이 추가되었습니다. 상세정보를 입력할 차례입니다. Author Certificate 오른쪽에 Generate 버튼을 클릭합니다.



Certificate Generator 팝업창이 나타나면 Key filename 속성에 파일명을 입력하고 Alias 속성에 소속단체명을 입력하고 비밀번호와 비밀번호 확인을 각각 입력합니다. Key filename과 Alias에 Profiles 이름과 동일하게 지정해도 상관없습니다. 여기까지만 입력해도 사용하는데는 문제가 없습니다. 입력이 끝났으면 OK 버튼을 누르고 빠져나가면 됩니다.

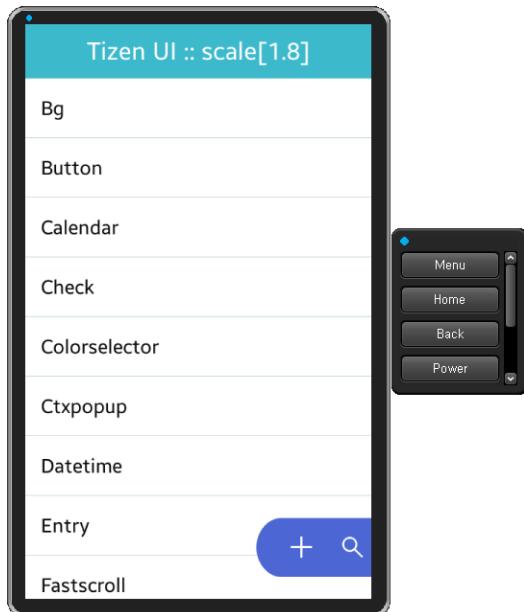


자동으로 매번 이 인증서를 사용할 것인지를 묻는 팝업창이 나타납니다.
Yes 버튼을 선택하면 됩니다.



이제 팝업창에서 Apply 버튼과 OK 버튼을 차례로 누르면 팝업창이 사라집니다. Signing 팝업창에서 OK 버튼을 클릭하면 소스 프로젝트가 에뮬레이터에 설치됩니다. 나중에 인증서를 수정할 필요가 있다면 이클립스 메인메뉴에서 [Windows > Preferences] 를 선택하고 Preferences 팝업창이 나타나면 왼쪽 트리목록에서 [Tizen SDK > Security Profiles]를 선택하면 됩니다.

에뮬레이터에 앱 설치가 진행되고 끝나면 UIControls가 실행됩니다.
UIControls 타이젠에서 지원하는 모든 위젯과 컨테이너의 사용 예시를
보여주는 예제입니다.

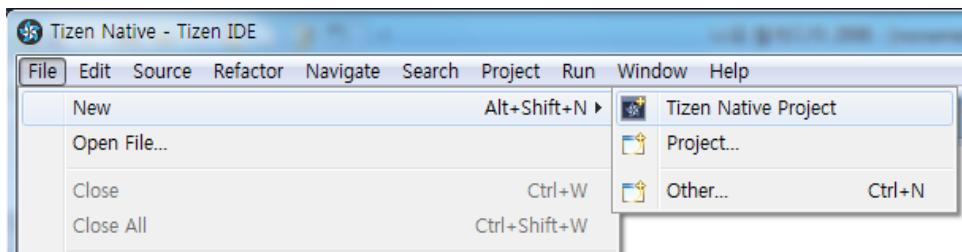


5. BasicUI 예제 만들기

BasicUI 방식으로 새로운 소스 프로젝트를 생성해서 Hello World라는 텍스트를 출력하는 예제를 만들어 보도록 하겠습니다.

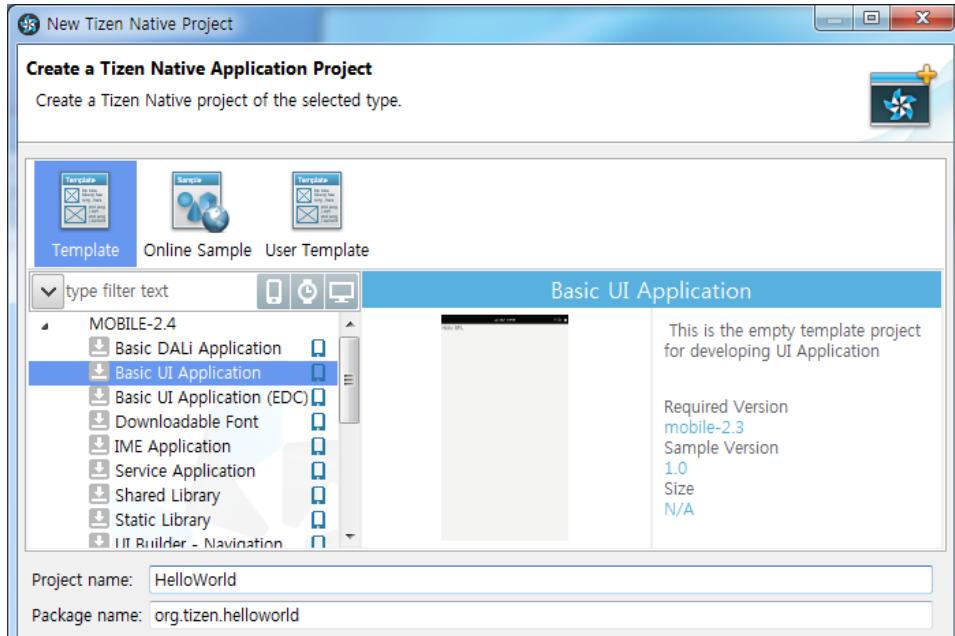
1) 소스 프로젝트 생성

새로운 소스 프로젝트를 생성하겠습니다. 이클립스 메인메뉴 [File > New > Tizen Native Project]를 선택합니다.



소스 프로젝트 생성 팝업창이 나타나면 [Template > MOBILE-2.x > Basic UI Application]을 선택합니다. 본 교재에서 이후의 예제는 모두 이 방식으로 생성하면 됩니다.

Project name 항목에는 HelloWorld라고 입력합니다.



Package name은 자동으로 입력됩니다. 만약 실제로 앱스토어에 등록할 앱을 개발한다면 Package name은 웹사이트 주소를 사용하면 됩니다. 예를 들어서 보유하고 있는 도메인 주소가 www.abc.com이라면 Package name을 com.abc.helloworld라고 입력하면 되는 것입니다.

이제 Finish 버튼을 누르면 새로운 소스 프로젝트가 생성됩니다.

2) 소스 프로젝트 구성요소

기본 소스 프로젝트에 대해서 간략하게 설명 드리겠습니다. Includes 폴더에는 라이브러리가 포함되어 있습니다.

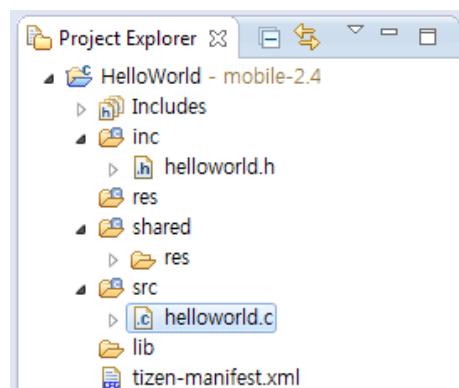
inc 폴더에는 C언어 헤더파일(~.h) 이 저장되어 있습니다. 라이브러리를 선언하거나 함수 헤더 혹은 전역 변수를 선언할 때 주로 사용합니다.

res 폴더에는 이미지나 오디오 파일 같은 리소스 파일을 주로 저장합니다.

src 폴더에는 C언어 소스파일(~.c)이 저장되어 있습니다. 함수의 기능을 정의할 때 주로 사용하며 대부분의 작업이 여기에서 이루어 집니다.

shared 폴더에는 앱 아이콘 이미지가 저장되어 있습니다. 앱스토어에 앱을 배포할 때 여기에 앱 아이콘을 저장하면 됩니다. 타이젠 스토어는 동그란 모양의 앱아이콘을 사용해야 합니다.

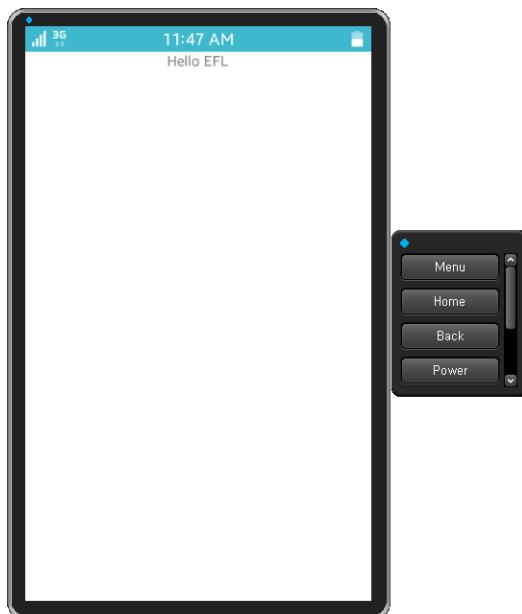
tizen-manifest.xml 파일은 앱의 각종 정보(앱 이름, 버전)와 사용자 권한(Privilege)를 저장하고 있습니다. 안드로이드의 AndroidManifest.xml과 동일한 기능입니다.



3) 기본 소스 프로젝트 실행

처음 소스 프로젝트가 생성된 상태로 실행시켜 보겠습니다. HelloWorld 프로젝트를 마우스 오른쪽 버튼으로 클릭하고, 단축메뉴에서 [Build Project] 를 선택합니다. 빌드가 끝나면 다시 HelloWorld 프로젝트를 마우스 오른쪽 버튼으로 클릭하고, 단축메뉴에서 [Run As > 1 Tizen Native Application]을 선택합니다.

에뮬레이터에 예제가 실행되면 위쪽에 인디케이터가 보이고, 그 아래에 앱 화면이 보입니다. 그리고 Hello EFL 이라는 글자가 보입니다. Label 위젯을 사용해서 텍스트를 표시한 것입니다.



4) 기본 소스 코드

Label 위젯에 표시된 Hello EFL이라는 텍스트를 변경해 보겠습니다.
그러기 위해서 소스파일을 편집해야 합니다. src 폴더를 열고
helloworld.c 파일을 더블클릭 합니다. 이클립스 메인화면에 파일 내용이
보입니다.

```
-----  
#include "helloworld.h"  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
} appdata_s;  
  
static void  
win_delete_request_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    ui_app_exit();  
}  
  
static void  
win_back_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    /* Let window go to hide state. */  
    elm_win_lower(ad->win);  
}
```

#include "helloworld.h"는 inc 폴더에 있는 helloworld.h를 참조한다는 의미입니다. 일반적으로 헤더파일에는 라이브러리 인클루드와 전역변수 & 함수 헤더를 선언하고, 소스파일에는 함수 내용을 정의합니다.

appdata_s는 앱에서 사용하는 데이터를 저장하는 구조체 입니다.

win_delete_request_cb()는 앱 삭제 요청이 발생했을 때 실행되는 이벤트 함수입니다. 이 함수를 직접 호출하지는 않습니다.

win_back_cb()는 Back 버튼을 눌렀을 때 실행되는 이벤트 함수입니다. 이 함수를 직접 호출하지는 않습니다.

아래쪽으로 내려가면 create_base_gui()라는 함수가 나옵니다. 화면을 구성하는 윈도우와 각종 컨테이너, 위젯을 생성하는 함수 입니다.

```
static void
create_base_gui(appdata_s *ad)
{
    /* Window */
    ad->win = elm_win_util_standard_add(PACKAGE, PACKAGE);
    elm_win_autodel_set(ad->win, EINA_TRUE);

    if (elm_win_wm_rotation_supported_get(ad->win)) {
        int rots[4] = { 0, 90, 180, 270 };
        elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots), 4);
    }

    evas_object_smart_callback_add(ad->win, "delete,request", win_delete_request_cb,
NULL);
    eext_object_event_callback_add(ad->win, EEXT_CALLBACK_BACK, win_back_cb, ad);

    /* Conformant */
}
```

```

ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);

/* Show window after base gui is set up */
evas_object_show(ad->win);
}

```

주요 코드 몇줄만 설명 드리겠습니다.

`elm_win_util_standard_add()` 는 Window 객체를 생성하는 API 입니다. Window는 화면 레이아웃 최상위 오브젝트 입니다. 하나의 앱은 하나의 Window를 가집니다. Window에 위젯을 배치하는 것도 가능하지만 컨테이너를 추가하고 그 컨테이너 위에 위젯을 추가하는 방식을 주로 사용합니다.

`elm_win_wm_rotation_available_rotations_set()` 는 화면 회전방향(Orientation)을 지정하는 API입니다. 4개의 각도(0, 90, 180, 270)를 배열로 전달하면 모든 화면 방향을 지원하게 됩니다.

`evas_object_smart_callback_add()` 는 위젯 혹은 컨테이너 같은 스마트 오브젝트에 이벤트 콜백 함수를 지정하는 API입니다. 1번째 파라미터에는 이벤트가 발생하는 오브젝트를 전달하고, 2번째 파라미터에는 이벤트 종류를 전달합니다. 3번째 파라미터에는 콜백 함수명을 전달하고, 4번째에는 사용자 데이터를 전달합니다. 이벤트 종류가 "delete,request" 이면 객체 삭제를 의미합니다.

`eext_object_event_callback_add()` 는 오브젝트에 이벤트 콜백 함수를 지정하는 API입니다. 스마트 오브젝트와 일반 오브젝트 모두에 사용 할 수 있습니다. 1번째 파라미터에는 이벤트가 발생하는 오브젝트를 전달하고, 2번째 파라미터에는 이벤트 종류를 전달합니다. 3번째 파라미터에는 콜백 함수명을 전달하고, 4번째에는 사용자 데이터를 전달합니다. EEXT_CALLBACK_BACK 는 Back 버튼 클릭 이벤트를 의미합니다.

`elm_conformant_add()` 는 Conformant를 생성하는 코드입니다. Conformant는 화면에 새로운 영역이 추가되었을 때(ex: 키패드) 윈도우 크기를 변경해주는 기능을 담당합니다. 하나의 앱은 하나의 Conformant만을 가져야 합니다. Conformant가 없어도 가능합니다. 화면 위쪽에 인디케이터(상태바)를 표시하려면 Conformant가 필요합니다. Conformant가 있을 때 인디케이터를 표시하지 않는 것도 가능합니다.

`elm_win_indicator_mode_set()` 는 인디케이터 표시 여부를 지정하는 API입니다.

`elm_win_indicator_opacity_set()` 는 인디케이터의 투명도를 지정하는 API입니다.

`evas_object_size_hint_weight_set()` 는 오브젝트의 크기를 대략적으로 지정하는 API입니다. 파라미터는 순서대로 오브젝트, 수평 크기 힌트, 수직 크기 힌트입니다. EVAS_HINT_EXPAND는 공간이 허락하는 만큼 최대한 크게 지정한다는 의미입니다.

`elm_win_resize_object_add()` 는 Window 오브젝트에 다른 오브젝트를 추가하면서 크기를 변경하는 API입니다.

`evas_object_show()` 는 오브젝트를 화면에 표시하는 API입니다. 오브젝트를 생성하면 디폴트 값이 Hide 상태입니다. 이 함수는 모든 오브젝트에 공통적으로 사용할 수 있습니다.

`elm_label_add()` 는 Label 위젯을 생성하는 API입니다. Label 위젯은 텍스트를 표시할 수 있으며 html 태그를 적용해서 텍스트 속성(폰트 크기, 컬러)을 변경할 수 있습니다.

5) Label 텍스트 변경

화면에 표시된 Hello EFL이라는 텍스트를 변경해 보겠습니다.

`elm_object_text_set()` 함수를 아래와 같이 변경합니다.

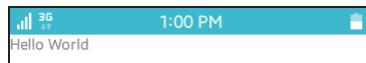
```
ad->label = elm_label_add(ad->conform);
//elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
elm_object_text_set(ad->label, "Hello World");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
```

`elm_object_text_set()` 는 위젯의 캡션 텍스트를 변경하는 API입니다.

Label 뿐만 아니라 Button, Entry에도 사용할 수 있습니다. 텍스트 내용에 Html 태그를 사용해서 텍스트 속성을 지정할 수도 있습니다.

다시 실행시켜 보겠습니다. 2번째로 실행할 때는 메인메뉴 [Run > Run]을 누르거나, 단축키 Ctrl + F11을 누르면 됩니다.

예제가 다시 실행되고 화면에 텍스트가 Hello World로 변경되었습니다.



6) 절대좌표를 이용한 Label 위치 이동

Label의 위치는 화면 좌측 위에 배치되어 있습니다.

evas_object_size_hint_weight_set() 함수는 오브젝트의 크기를 상대적으로 지정하는 함수인데, 수평, 수직 옵션이 모두 확장(EVAS_HINT_EXPAND)으로 지정되어 있기 때문입니다. 1번째 파라미터에는 속성을 지정할 오브젝트, 2번째는 수평 크기 힌트, 3번째는 수직 크기 힌트입니다.

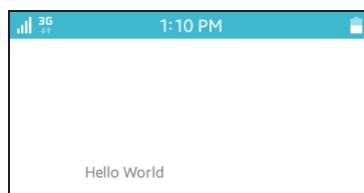
절대값을 지정해서 Label의 위치를 이동시켜 보겠습니다. 코드를 아래와 같이 변경합니다.

```
elm_object_text_set(ad->label, "Hello World");
//evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
//elm_object_content_set(ad->conform, ad->label);
evas_object_move(ad->label, 100, 200);
evas_object_resize(ad->label, 400, 100);
evas_object_show(ad->label);
```

`evas_object_move()` 는 오브젝트의 크기를 절대값으로 지정하는 함수입니다. 파라미터에 순서대로 속성을 지정할 오브젝트, X좌표 위치, Y좌표 위치입니다.

`evas_object_resize()`는 오브젝트의 위치를 절대값으로 지정하는 함수입니다. 파라미터에 순서대로 속성을 지정할 오브젝트, X좌표 넓이, Y좌표 높이 입니다.

앱을 다시 실행하면 Label의 위치가 달라져 있습니다.



7) Box 컨테이너를 이용한 해상도 호환

위와 같이 절대좌표를 지정하면 편리하기는 하지만 다양한 단말의 해상도를 지원할수 없습니다. Box 컨테이너를 이용해서 다양한 해상도를 지원할 수 있는 방법을 알아보겠습니다.

`create_base_gui()` 함수 위에 아래와 같이 새로운 함수를 추가하겠습니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",
```

```

* "pad_large" and "pad_huge" available as styles in addition to the
* "default" frame style */
elm_object_style_set(frame, "pad_medium");
/* set the input weight/aling on the frame insted of the child */
evas_object_size_hint_weight_set(frame, h_weight, v_weight);
evas_object_size_hint_align_set(frame, h_align, v_align);
{
    /* tell the child that is packed into the frame to be able to expand */
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    /* fill the expanded area (above) as opposaed to center in it */
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    /* actually put the child in the frame and show it */
    evas_object_show(child);
    elm_object_content_set(frame, child);
}
/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}

```

my_box_pack() 는 Box 컨테이너 안에 위젯을 추가하는 함수입니다. Box는 수직 혹은 수평 방향으로 위젯 혹은 컨테이너를 순차적으로 추가하는 컨테이너입니다. 안드로이드의 LinearLayout과 유사한 역할을 수행합니다. 앞으로 이 함수는 여러 예제에서 자주 사용될 것입니다. Box 컨테이너에 대한 자세한 설명은 Box 예제에서 설명 드리겠습니다.

my_box_pack() 함수의 1번째 파라미터는 Box 컨테이너의 핸들이고, 2번째 파라미터는 Box 컨테이너에 추가되는 위젯 혹은 컨테이너입니다. 3번째 파라미터는 위젯이 차지하는 수평 영역의 힌트를 지정합니다. EVAS_HINT_EXPAND 혹은 1.0은 최대한 크게 지정하고, 0.0은 최소한의

영역만 지정하게 됩니다. Bg 위젯의 경우에는 기본 사이즈가 0 이므로 0.0을 지정하게 되면 화면에 나타나지 않습니다.

4번째 파라미터는 위젯이 차지하는 수직 영역의 힌트를 지정합니다.

5번째 파라미터는 위젯의 수평 위치를 지정합니다. 0.0이면 왼쪽 정렬, 0.5이면 중앙 정렬, 1.0이면 오른쪽 정렬, EVAS_HINT_FILL 혹은 -1 이면 수평 영역 전체를 채웁니다.

6번째 파라미터는 위젯의 수직 위치를 지정합니다. 0.0이면 위쪽 정렬, 0.5이면 가운데 정렬, 1.0이면 아래쪽 정렬, EVAS_HINT_FILL 혹은 -1 이면 수직 영역 전체를 채웁니다.

create_base_gui() 함수의 아래쪽 소스코드를 아래와 같이 수정합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
    /* A box to put things in vertically - default mode for box */
    Evas_Object *box = elm_box_add(ad->win);
    evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    elm_object_content_set(ad->conform, box);
    evas_object_show(box);

}

/* Label*/
```

```
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "Hello World");
    my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);
}
}

/* Show window after base gui is set up */
evas_object_show(ad->win);
}
```

{ } 기호는 상위 윈도우와 하위 윈도우의 포함 관계를 쉽게 확인하기 위해서 표기하였습니다. 꼭 필요한 것은 아니지만 있을때와 없을때의 가독성 차이는 많기 때문에 귀찮더라도 사용하는 습관을 들이기 바랍니다.

elm_box_add()는 Box 컨테이너를 생성하는 API입니다.

예제를 다시 실행시켜 보겠습니다. 이전과 크게 달라진 것은 없지만 이제 다른 해상도의 단말에서 실행해도 동일한 위치에 위젯이 표시됩니다.



8) 비효율적인 코드 수정

소스 프로젝트를 생성했을 때 기본적으로 추가되는 소스 코드 중에서 불필요한 부분을 제거하면 성능을 더 향상시킬 수 있습니다. EFL을 개발한 Carsten Haitzler이 권장하는 방식으로 소스코드를 수정해 보겠습니다.

소스 파일 위쪽으로 이동해서 `win_delete_request_cb()` 함수를 삭제하고, `win_back_cb()` 함수의 내용을 아래와 같이 수정합니다.

```
/*static void
win_delete_request_cb(void *data, Evas_Object *obj, void *event_info)
{
    ui_app_exit();
}*/
```

```
static void
win_back_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    /* Let window go to hide state. */
    //elm_win_lower(ad->win);
    elm_win_iconified_set(ad->win, EINA_TRUE);
}
```

`win_delete_request_cb()` 함수는 `create_base_gui()` 함수의 아래 코드에서 사용하는 콜백 함수입니다. 이것은 PC에서 사용되는 이벤트라서 모바일에서는 별 의미가 없습니다.

```
evas_object_smart_callback_add(ad->win, "delete,request",
win_delete_request_cb, NULL);
```

create_base_gui() 함수로 이동해서 시작 부분을 아래와 같이 수정합니다.

```
static void
create_base_gui(appdata_s *ad)
{
    /* set up policy to exit when last window is closed */
    elm_policy_set(ELM_POLICY_QUIT, ELM_POLICY_QUIT_LAST_WINDOW_CLOSED);

    /* Window */
    ad->win = elm_win_util_standard_add(PACKAGE, PACKAGE);
    elm_win_autodel_set(ad->win, EINA_TRUE);

    int rots[4] = { 0, 90, 180, 270 };
    elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots), 4);

    eext_object_event_callback_add(ad->win, EEXT_CALLBACK_BACK, win_back_cb, ad);

    /* child object - indent to how relationship */
    /* Conformant */
    ad->conform = elm_conformant_add(ad->win);
    elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
    elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
    evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    elm_win_resize_object_add(ad->win, ad->conform);
    evas_object_show(ad->conform);
```

`elm_policy_set()` 함수를 사용해서 마지막 윈도우가 사라질 때 앱이 종료되도록 지정하였습니다.

`elm_win_wm_rotation_supported_get()` 함수는 모바일에서는 기종에 상관없이 지원되는 기능이라서 판단할 필요가 없기 때문에 제거하였습니다.

마지막으로 필요없는 이벤트 콜백 함수 호출 기능을 제거하겠습니다. 소스파일 아래쪽으로 이동해서 다음 4개의 콜백함수를 삭제합니다.

- `ui_app_orient_changed()`
- `ui_app_region_changed()`
- `ui_app_low_battery()`
- `ui_app_low_memory()`

그리고 가장 아래쪽에 있는 `main()` 함수에서 위 함수를 콜백함수를 지정하는 부분을 삭제합니다.

```
int
main(int argc, char *argv[])
{
    appdata_s ad = {0,};
    int ret = 0;

    ui_app_lifecycle_callback_s event_callback = {0,};
    app_event_handler_h handlers[5] = {NULL, };

    event_callback.create = app_create;
    event_callback.terminate = app_terminate;
    event_callback.pause = app_pause;
    event_callback.resume = app_resume;
    event_callback.app_control = app_control;
```

```

//ui_app_add_event_handler(&handlers[APP_EVENT_LOW_BATTERY],
APP_EVENT_LOW_BATTERY, ui_app_low_battery, &ad);
//ui_app_add_event_handler(&handlers[APP_EVENT_LOW_MEMORY],
APP_EVENT_LOW_MEMORY, ui_app_low_memory, &ad);

//ui_app_add_event_handler(&handlers[APP_EVENT_DEVICE_ORIENTATION_CHANGED],
APP_EVENT_DEVICE_ORIENTATION_CHANGED, ui_app_orient_changed, &ad);
    ui_app_add_event_handler(&handlers[APP_EVENT_LANGUAGE_CHANGED],
APP_EVENT_LANGUAGE_CHANGED, ui_app_lang_changed, &ad);
    //ui_app_add_event_handler(&handlers[APP_EVENT_REGION_FORMAT_CHANGED],
APP_EVENT_REGION_FORMAT_CHANGED, ui_app_region_changed, &ad);
    //ui_app_remove_event_handler(handlers[APP_EVENT_LOW_MEMORY]);

    ret = ui_app_main(argc, argv, &event_callback, &ad);
    if (ret != APP_ERROR_NONE) {
        dlog_print(DLOG_ERROR, LOG_TAG, "app_main() is failed. err = %d", ret);
    }

    return ret;
}

```

9) 관련 API

appdata_s : 앱 정보를 저장하는 구조체.

void create_base_gui(appdata_s *ad) : 화면을 구성하는 윈도우와 각종 컨테이너, 위젯을 생성하는 함수.

void win_delete_request_cb() : 앱 삭제 요청이 발생했을 때 실행되는 이벤트 함수. 이 함수를 직접 호출하지는 않습니다.

`void win_back_cb()` : Back 버튼을 눌렀을 때 실행되는 이벤트 함수. 이 함수를 직접 호출하지는 않습니다.

`Evas_Object *elm_win_util_standard_add(char *name, char *title)` : Window 객체 생성 함수. Window는 화면 레이아웃 최상위 오브젝트입니다. 하나의 앱은 하나의 Window 를 가집니다. Window에 위젯을 배치하는 것도 가능하지만 주로 컨테이너를 추가하고, 그 컨테이너 위에 위젯을 추가하는 방식을 사용합니다.

`void elm_win_wm_rotation_available_rotations_set(Elm_Win *obj, const int *rotations, unsigned int count)` : 화면 회전방향(Orientation)을 지정하는 API. 4개의 각도(0, 90, 180, 270)를 배열로 전달하면 모든 화면 방향을 지원하게 됩니다.

`void evas_object_smart_callback_add(Evas_Object *obj, const char *event, Evas_Smart_Cb func, const void *data)` : 위젯 혹은 컨테이너 같은 스마트 오브젝트에 이벤트 콜백 함수를 지정하는 API. 1번째 파라미터에는 이벤트가 발생하는 오브젝트를 전달하고, 2번째 파라미터에는 이벤트 종류를 전달합니다. 3번째 파라미터에는 콜백 함수명을 전달하고, 4번째에는 사용자 데이터를 전달합니다. 이벤트 종류가 "delete,request" 이면 객체 삭제를 의미합니다.

`void eext_object_event_callback_add(Evas_Object *obj, Eext_Callback_Type type, Eext_Event_Cb func, void *data)` : 오브젝트에 이벤트 콜백 함수를 지정하는 API. 스마트 오브젝트와 일반 오브젝트 모두에 사용 할 수 있습니다. 1번째 파라미터에는 이벤트가 발생하는 오브젝트를 전달하고, 2번째 파라미터에는 이벤트 종류를 전달합니다. 3번째 파라미터에는 콜백 함수명을 전달하고, 4번째에는 사용자 데이터를 전달합니다. EEXT_CALLBACK_BACK 는 Back 버튼 클릭 이벤트를 의미합니다.

Evas_Object *elm_conformant_add(Evas_Object *parent) : Conformant 컨테이너 생성 함수. Conformant 는 화면에 새로운 영역이 추가되었을 때(키패드) 윈도우 크기를 변경해주는 기능을 담당합니다. 하나의 앱은 하나의 Conformant 만을 가져야 합니다. 혹은 Conformant가 없어도 가능합니다. 화면 위쪽에 인디케이터(상태바)를 표시하려면 Conformant가 필요합니다. Conformant가 있을 때 인디케이터를 표시하지 않는 것도 가능합니다.

void elm_win_indicator_mode_set(Elm_Win *obj, Elm_Win_Indicator_Mode mode) : 인디케이터 표시 여부를 지정하는 API.

void elm_win_indicator_opacity_set(Elm_Win *obj, Elm_Win_Indicator_Opacity_Mode mode) : 인디케이터의 투명도를 지정하는 API.

void evas_object_size_hint_weight_set(Evas_Object *obj, double x, double y) : 오브젝트의 크기를 대략적으로 지정하는 API. 파라미터는 순서대로 오브젝트, 수평 크기 힌트, 수직 크기 힌트입니다. EVAS_HINT_EXPAND 는 공간이 허락하는 만큼 최대한 크게 지정한다는 의미입니다.

void elm_win_resize_object_add(Elm_Win *obj, Evas_Object *subobj) : Window 오브젝트에 다른 오브젝트를 추가하면서 크기를 변경하는 API.

Evas_Object *elm_label_add(Evas_Object *parent) : Label 위젯 생성 함수. 텍스트를 표시할 수 있으며 html 태그를 적용해서 텍스트 속성(폰트 크기, 컬러)을 변경할 수 있습니다.

void evas_object_show(Evas_Object *obj) : 오브젝트를 화면에 표시하는 함수. 오브젝트를 생성하면 디폴트 값이 Hide 상태입니다.

`evas_object_show()` 함수는 모든 오브젝트에 공통적으로 사용할 수 있습니다.

`void elm_object_text_set(Evas_Object *obj, const char *text)` : 위치의 캡션 텍스트를 변경하는 함수. Label 뿐만 아니라 Button, Entry에도 사용할 수 있습니다.

`void evas_object_move(Evas_Object *obj, Evas_Coord x, Evas_Coord y)` : 오브젝트의 크기를 절대값으로 지정하는 함수. / 파라미터 : 속성을 지정할 오브젝트, X좌표 위치, Y좌표 위치.

`void evas_object_resize(Evas_Object *obj, Evas_Coord w, Evas_Coord h)` : 오브젝트의 위치를 절대값으로 지정하는 함수. / 파라미터 : 속성을 지정할 오브젝트, 넓이, Y좌표 높이.

6. Label 위젯 사용방법

텍스트를 화면에 표시하려면 Label 위젯을 사용하면 됩니다. html 태그를 적용해서 각종 속성(폰트 크기, 컬러)을 변경할 수 있습니다.

1) Label 텍스트 가운데 정렬

새로운 소스 프로젝트를 생성하고 Project name을 LabelEx으로 지정합니다. 방법은 이클립스 메인 메뉴에서 [File > New > Tizen Native Project]를 선택하고, 팝업창이 나타나면 [Template > MOBILE-2.4 > Basic UI Application}을 선택하면 됩니다.

소스 프로젝트가 생성되었으면 src 폴더에 labelex.c 파일을 열고 create_base_gui() 함수로 이동해서 Label 위젯 생성 코드를 다음과 같이 수정합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Label-1 */
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
//evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
```

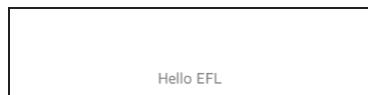
```
EVAS_HINT_EXPAND);  
//elm_object_content_set(ad->conform, ad->label);  
evas_object_move(ad->label, 120, 80);  
evas_object_resize(ad->label, 240, 80);  
evas_object_show(ad->label);
```

elm_object_text_set() 은 위젯의 캡션 텍스트를 지정하는 API 입니다.
2번째 파라미터에 Html 태그를 입력하면 텍스트에 속성이 적용됩니다.
<align=center> 는 가운데 정렬을 의미합니다.

evas_object_move() 는 위젯의 좌측 상단 시작 위치를 지정하는 API 입니다. 2번째 파라미터는 x좌표, 3번째 파라미터는 y좌표 입니다.

evas_object_resize() 는 위젯의 크기를 지정하는 API 입니다. 2번째 파라미터는 넓이, 3번째 파라미터는 높이 입니다.

소스 프로젝트를 빌드하고 실행하면 Hello EFL이라는 텍스트 표시됩니다.
수평방향으로 가운데에 위치하고 있습니다.



2) 폰트 크기 변경

Label 위젯 캡션 텍스트의 폰트 크기를 변경해 보겠습니다.
create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다. 2번째 Label 위젯을 생성하는 코드입니다.

```

/* Label-1 */
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
//evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
//elm_object_content_set(ad->conform, ad->label);
evas_object_move(ad->label, 120, 80);
evas_object_resize(ad->label, 240, 80);
evas_object_show(ad->label);

/* Label-2 */
Evas_Object *label = elm_label_add(ad->conform);
elm_object_text_set(label, _("<font_size=20><align=center>fontsize is set to
20</align></font_size>"));
evas_object_move(label, 120, 160);
evas_object_resize(label, 240, 80);
evas_object_show(label);

/* Show window after base gui is set up */
evas_object_show(ad->win);

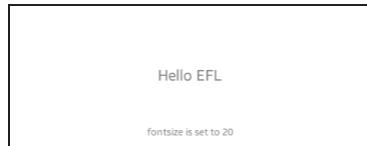
```

Evas_Object는 위젯 혹은 컨테이너처럼 화면에 배치되는 오브젝트의 공통 변수입니다. 그래서 Conformant와 Label을 동일한 변수 타입으로 선언해도 상관 없습니다.

`elm_object_text_set()` 은 캡션 텍스트를 변경하는 함수입니다. Label 위젯 뿐만 아니라 Button, Entry에도 사용할 수 있습니다. 이 함수에 html 태그를 대입하면 웹 브라우저에서 보이는 것과 동일한 효과가 나타납니다.

`<font_size=20>` 은 폰트 크기를 20 픽셀로 지정하는 태그입니다.
`<align=center>` 수평 정렬 위치를 중앙으로 지정하는 태그입니다.

예제를 다시 실행하면 2번째 Label 위젯이 추가되었고 'fontsize is set to 20'이라는 텍스트가 표시됩니다. 1번째 Label 보다 텍스트 크기가 작아졌습니다. 수평 위치는 가운데 정렬 방식이 적용되었습니다.



3) 폰트 컬러 변경

Label 위젯의 폰트 컬러를 변경해 보겠습니다. 폰트 크기와 마찬가지로 html 태그를 사용하면 됩니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
    evas_object_show(label);

    /* Label-3 */
    label = elm_label_add(ad->conform);
    elm_object_text_set(label, _("<color=#FF4500FF><align=right>font color orange
red</align></color>"));
    evas_object_move(label, 50, 240);
    evas_object_resize(label, 380, 80);
    evas_object_show(label);

    /* Show window after base gui is set up */
    evas_object_show(ad->win);
```

`elm_object_text_set()` 함수에 입력된 `<color=#FF4500FF>` 는 컬러를 지정하는 태그입니다. AARRGGBB 형식을 사용하였습니다. 반투명도는 최대치라서 불투명이고, 파란색은 45, 초록색은 0, 빨간색은 최대치입니다. `<color=#F40F>`를 입력해도 결과는 동일합니다.

예제를 다시 실행하면 3번째 Label 위젯이 추가되었고 오렌지색 컬러의 텍스트가 표시됩니다.



4) 말 줄임표

텍스트 내용이 길어서 오른쪽 끝영역을 벗어날때, 말줄임표를 표시하는 방법을 알아보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
    evas_object_show(label);

/* Label-4 */
label = elm_label_add(ad->conform);
elm_object_text_set(label, _("<font_size=24>If the string length exceeds the
width</font_size>"));
evas_object_move(label, 72, 320);
evas_object_resize(label, 560, 80);
elm_label_ellipsis_set(label, EINA_TRUE);
evas_object_show(label);
```

```
/* Show window after base gui is set up */
evas_object_show(ad->win);
```

elm_label_ellipsis_set() 은 Label 위젯에 말줄임표를 적용하는 API입니다.
1번째 파라미터에는 속성을 지정할 Label 위젯을 전달하고, 2번째
파라미터에는 true 혹은 false를 전달합니다. EINA_TRUE는 타이젠에서
사용하는 boolean 타입의 true 값입니다. false를 지정하려면
EINA_FALSE를 사용하면 됩니다.

예제를 다시 실행하면 4번째 텍스트가 표시되고,
오른쪽 끝에 말줄임표가 표시됩니다.



5) 멀티 라인 텍스트

Label 위젯에 여러 줄의 텍스트를 표시하려면
 태그를 사용하면 됩니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
evas_object_show(label);

/* Label-5 */
label = elm_label_add(ad->conform);
elm_label_line_wrap_set(label, EINA_TRUE);
elm_object_text_set(label, _("<font_size=20><align=left>Once upon a time
there lived a young prince.<br>Mountain is mountain, water is water.
```

```

</align></font_size>"));
    evas_object_move(label, 120, 400);
    evas_object_resize(label, 240, 160);
    evas_object_show(label);

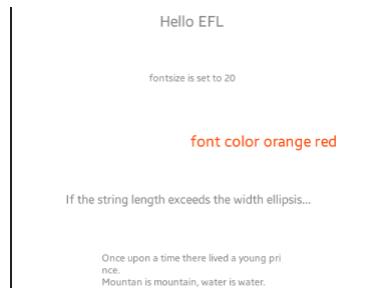
/* Show window after base gui is set up */
evas_object_show(ad->win);

```

`elm_label_line_wrap_set()` 은 Label 위젯에 자동줄바꿈을 지정하는 API입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 자동 줄바꿈이 적용됩니다.

5번째 Label 위젯을 생성하였고, 장문의 텍스트를 입력하였습니다.
 태그를 사용해서 2줄로 분할 하였습니다.

예제를 다시 실행하면 5번째 텍스트가 표시되고,
 태그가 있는 위치에서 줄이 바뀝니다.



6) 관련 API

Evas_Object : 위젯 혹은 컨테이너와 같은 화면에 배치되는 오브젝트의 공통 변수. 그래서 Conformant와 Label 처럼 서로 다른 오브젝트를 동일한 변수 타입으로 선언해도 됩니다.

void elm_object_text_set(Evas_Object *obj, char *text) : 캡션 텍스트를 변경하는 API. Label 위젯 뿐만 아니라 Button, Entry에도 사용할 수 있습니다. 이 함수에 html 태그를 대입하면 웹 브라우저에서 보이는 것과 동일한 효과가 나타납니다.

EINA_TRUE : 타이젠에서 사용하는 boolean 타입의 true 값.

EINA_FALSE : 타이zen에서 사용하는 boolean 타입의 false 값.

void elm_label_ellipsis_set(Evas_Object *obj, Eina_Bool ellipsis) : Label 위젯의 캡션 텍스트가 오른쪽 끝영역을 벗어날 경우에 말줄임표를 표시하는 API. 2번째 파라미터에 EINA_TRUE를 입력하면 말줄입표가 표시되고, EINA_FALSE를 입력하면 해제됩니다.

7. Button 위젯 사용방법

Button 위젯은 사용자의 입력을 받을 수 있으며 가장 자주 사용되는 위젯입니다. 터치 이벤트를 구할 수 있고, EDJE를 사용해서 배경 이미지를 적용할 수 있습니다.

1) Label 위젯 텍스트 변경

새로운 소스 프로젝트를 생성하고 Project name을 ButtonEx으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 buttonex.c 파일을 열고 create_base_gui() 함수 위에 새로운 함수를 추가합니다. HelloWorld 예제와 동일한 함수입니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame instead of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    /*
     * tell the child that is packed into the frame to be able to expand */
}
```

```

evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

/* fill the expanded area (above) as opposed to center in it */
evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
/* actually put the child in the frame and show it */
evas_object_show(child);
elm_object_content_set(frame, child);

}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);

}

```

그런 다음 `create_base_gui()` 함수로 이동해서 Label 위젯을 생성하는 코드를 수정합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
    Evas_Object *box = elm_box_add(ad->win);
    evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    elm_object_content_set(ad->conform, box);
    evas_object_show(box);
}
```

```

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "<align=center>Press a Button</>");
    //evas_object_size_hint_weight_set(ad->label,      EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    //elm_object_content_set(ad->conform, ad->label);
    my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);
}
}

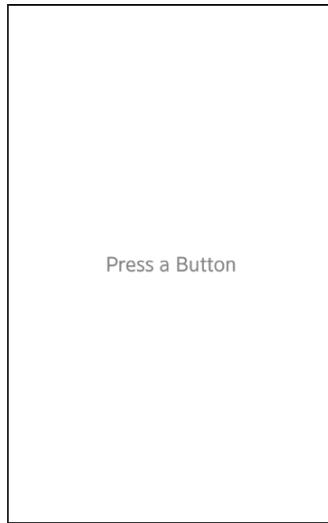
/* Show window after base gui is set up */
evas_object_show(ad->win);
}

```

my_box_pack() 함수의 파라미터는 순서대로 다음과 같습니다.

- Box 컨테이너
- 자식 윈도우
- 수평 넓이 힌트 : 1.0 이면 최대, 0.0이면 최소
- 수직 높이 힌트 : 1.0 이면 최대, 0.0이면 최소
- 수평 위치 : 0.0이면 왼쪽, 0.5이면 중앙, 1.0이면 오른쪽, -1이면 가득
채우기
- 수직 위치 : 0.0이면 위쪽, 0.5이면 가운데, 1.0이면 아래쪽, -1이면 가득
채우기

예제를 실행하면 화면에 Press Button이라는 텍스트가 표시됩니다.



2) Button 위젯 생성

create_base_gui() 함수 끝부분에 Button 위젯을 생성하는 코드를 추가하겠습니다.

```
{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "<align=center>Press a Button</>");
    my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

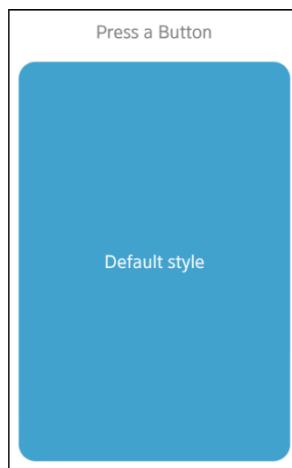
    /* Button-1 */
    Evas_Object *btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "Default style");
    my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
}
```

elm_button_add() 는 Button 위젯을 생성하는 API입니다.

`elm_object_text_set()` 함수를 사용해서 캡션 텍스트를 'Default style'로 지정합니다.

그외 함수는 이전 예제에서 살펴 보았던 종류들입니다.

소스 프로젝트를 빌드하고 실행하면 Button 위젯이 추가되었고 'Default style'이라는 텍스트가 표시됩니다. Button을 클릭하면 아무런 변화가 없습니다. 이벤트 함수를 정의하지 않았기 때문입니다.



3) Button 이벤트 함수 정의

타이젠에서 사용하는 EFL 라이브러리는 콜백 스타일로 이벤트 함수를 지정합니다. 웹 프로그래밍을 경험해 보셨다면 익숙한 방식일 것입니다. Button 이벤트 함수를 정의하겠습니다. `btn_default_cb()`라는 이름의 함수를 추가합니다.

주의할 것은 이 함수를 `create_base_gui()` 함수에서 호출하기 때문에 `btn_default_cb()` 가 `create_base_gui()` 보다 앞쪽에 위치해야 한다는 것입니다. 헤더파일에 함수 헤더를 선언해 두면 순서에 상관없이

서로서로 호출할 수 있습니다.

```
static void  
btn_default_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s* ad = data;  
    elm_object_text_set(obj, "Button Pressed");  
    elm_object_text_set(ad->label, "Button-1 Pressed");  
}
```

```
static void  
create_base_gui(appdata_s *ad)  
{
```

btn_default_cb() 함수는 3개의 파라미터를 받습니다. 1번째는 호출하는 쪽에서 전달해주는 사용자 데이터입니다. 여기서는 앱데이터(appdata_s)를 사용할 것입니다. 2번째 파라미터는 이벤트가 발생한 오브젝트입니다. 여기서는 1번째 Button입니다. 3번째는 이벤트 정보를 담고 있는 구조체입니다.

elm_object_text_set() 는 위젯의 캡션 텍스트를 변경하는 API입니다.

elm_object_text_set() 는 위젯의 캡션 텍스트를 변경하는 API입니다.

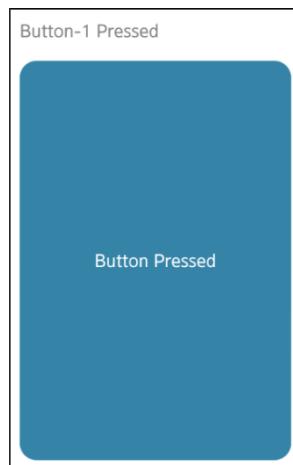
이제 Button을 생성하는 코드로 이동해서 위 함수를 콜백 함수로 지정해 주면 됩니다. create_base_gui() 함수로 이동해서 한줄의 코드를 추가 하겠습니다.

```
/* Button-1 */  
Evas_Object *btn = elm_button_add(ad->conform);
```

```
elm_object_text_set(btn, "Default style");
evas_object_smart_callback_add(btn, "clicked", btn_default_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
```

evas_object_smart_callback_add() 는 컨테이너 혹은 위젯 같은 스마트 오브젝트에 콜백 이벤트 함수를 지정하는 함수입니다. evas 오브젝트는 화면에 그려지는 모든 오브젝트를 의미합니다. 스마트 오브젝트는 Evas에서 제공하는 기본 오브젝트(Line, Rect, Polygon, Text, Image) 외에 추가 오브젝트를 의미합니다. 1번째 파라미터는 이벤트가 발생하는 오브젝트입니다. 2번째 파라미터는 이벤트 종류를 지정합니다. "clicked"는 클릭 이벤트를 의미합니다. 3번째 파라미터는 콜백 함수명을 지정합니다. 4번째 파라미터는 콜백 함수에 전달하는 데이터입니다. 여기서는 앱데이터 입니다.

예제를 다시 실행하고 Button을 클릭해 봅시다. Label 과 Button에 텍스트가 변경됩니다.



3) Button 아이콘 적용 - 재요청

Button에는 몇가지 아이콘 이미지를 표시할 수 있습니다. 하나씩 사용방법을 알아보도록 하겠습니다.

create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다. 2번째 Button을 생성하는 코드입니다.

```
/* Button-1 */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Default style");
evas_object_smart_callback_add(btn, "clicked", btn_default_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_style_set(btn, "icon_reorder");
evas_object_smart_callback_add(btn, "clicked", btn_icon_reorder_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
```

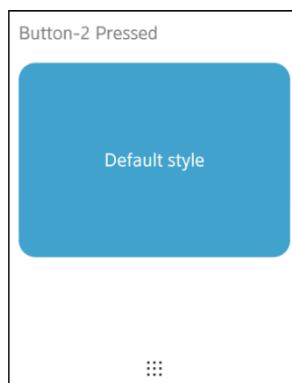
elm_object_style_set() 는 위젯의 스타일을 지정하는 함수입니다. 1번째 파라미터는 스타일이 적용되는 위젯입니다. 2번째 파라미터는 스타일 종류를 지정합니다. 'icon_reorder' 는 재요청 아이콘을 표시하라는 의미입니다.

evas_object_smart_callback_add()에 btn_icon_reorder_cb 라는 콜백 함수를 지정했습니다. 이 함수는 아직 구현되지 않았습니다.
create_base_gui() 함수 위에 아래와 같이 새로운 코드를 추가합니다.

```
static void  
btn_icon_reorder_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s* ad = data;  
    elm_object_text_set(ad->label, "Button-2 Pressed");  
}
```

2번째 Button의 콜백 함수를 정의했고, Label 위젯에 텍스트를 변경하는 코드를 추가했습니다.

예제를 다시 실행하면 아이콘 이미지가 표시된 2번째 버튼이 추가되었습니다. 2번째 Button을 클릭해 봅시다. Label과 2번째 Button에 텍스트가 변경됩니다.



4) Button 아이콘 적용 - 플러스 & 마이너스

이번에는 플러스 아이콘과 마이너스 아이콘이 적용된 Button을 생성해 보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다. 3번째 Button을 생성하는 코드입니다.

```
/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_style_set(btn, "icon_reorder");
evas_object_smart_callback_add(btn, "clicked", btn_icon_reorder_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

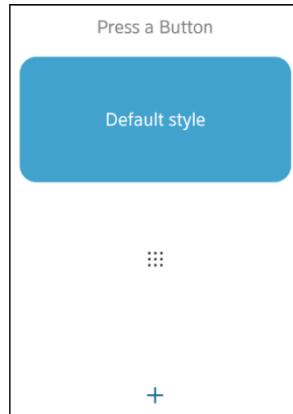
/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_style_set(btn, "icon_expand_add");
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
```

`elm_object_style_set()` 함수의 2번째 파라미터에 '`icon_expand_add`'를 지정하면 플러스 아이콘 이미지가 표시됩니다.

콜백 함수를 지정하는 방법은 이전과 동일하기 때문에 생략하였습니다.

예제를 다시 실행하면 3번째 버튼에 플러스 아이콘이 표시되어 있습니다.



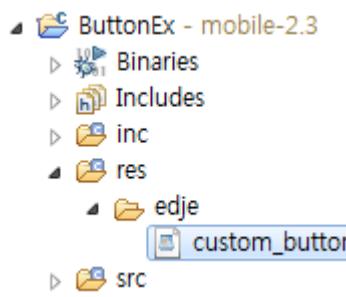
5) Button에 배경 이미지 적용

Button 위젯에 배경 이미지를 적용하려면 테마를 사용해야 합니다. EFL에서는 EDJE라는 테마를 제공하고 있습니다. 테마 파일을 처음부터 새로 만드는 것은 번거로우므로 부록에서 가져오도록 하겠습니다.

소스 프로젝트 /res 폴더에 새로운 폴더를 생성하고 이름을 edje 라고 지정합니다. 방법은 /res 폴더를 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [New > Folder]를 선택합니다. 팝업창이 나타나면 Folder name 항목에 edje라고 입력하고 Finish 버튼을 누르면 됩니다.

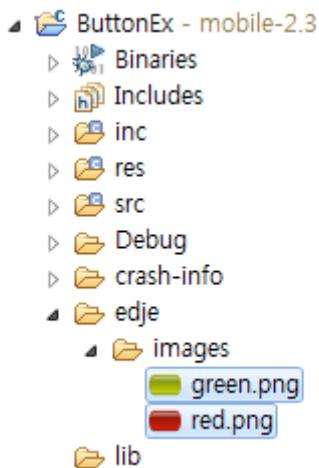
부록 /etc/edje 폴더에 들어가면 custom_button.edc라는 파일이 있는데 이것을 방금 생성한 /res/edje 폴더로 복사합니다.

res 폴더를 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [New > Folder]를 선택하면 새로운 폴더를 생성할 수 있습니다. 파일을 마우스 드래그하면 원하는 폴더로 복사할 수 있습니다.



Button에 배경 이미지를 적용하려면 이미지 파일이 필요합니다. 배경 이미지로 사용할 이미지 파일을 복사하겠습니다. 소스 프로젝트 루트 폴더에 새로운 폴더를 생성하고 이름을 edje라고 지정합니다. 그런 다음 /edje 폴더 안에 새로운 폴더를 생성하고 이름을 images라고 지정합니다.

부록 /Image 폴더에 들어가서 green.png와 red.png 2개 파일을 방금 생성한 /edje/images 폴더로 복사합니다.



소스코드를 추가할 차례입니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다. EDJE 파일을 테마로 등록하고 4번째 Button 위젯을 생성하는 코드입니다.

```

/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_style_set(btn, "icon_expand_add");
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Theme */
char edj_path[PATH_MAX] = "";
app_get_resource("edje/custom_button.edj", edj_path, (int)PATH_MAX);
elm_theme_extension_add(NULL, edj_path);

/* Button-4 */
btn = elm_button_add(ad->win);
elm_object_style_set(btn, "customized");
elm_object_text_set(btn, "Custom style");
evas_object_smart_callback_add(btn, "clicked", btn_custom_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
}

```

`app_get_resource()` 는 `/res` 폴더의 절대경로를 구해서 하위 폴더 경로를 추가한 다음 반환해주는 함수입니다. 1번째 파라미터에는 하위 경로를 지정하는데 여기서는 '`edje/custom_button.edj`'을 전달했기 때문에 전체 경로는 `~/res/edje/custom_button.edj`가 됩니다. 이 함수는 아직 생성되지 않았습니다. 잠시 후에 작성해 보도록 하겠습니다.

`elm_theme_extension_add()` 는 테마 정보 파일을 등록하는 함수입니다. 2번째 파라미터에 EDJE 파일의 경로를 전달하면 됩니다.

`elm_object_style_set()` 는 Button 위젯에 커스텀 테마를 적용하는 코드입니다. 커스텀 테마의 이름은 '`customized`'라고 지정하였습니다.

이 테마는 custom_button.edc 파일에 정의되어 있습니다.

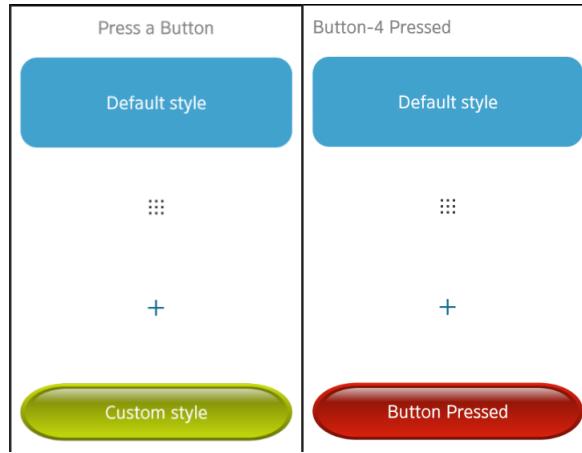
create_base_gui() 함수 위에 2개의 함수를 추가하겠습니다. 1번째는 5번째 버튼의 콜백 이벤트 함수이고, 2번째는 /res 폴더의 절대경로를 반환하는 함수입니다.

```
static void
btn_custom_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s* ad = data;
    elm_object_text_set(obj, "Button Pressed");
    elm_object_text_set(ad->label, "Button-5 Pressed");
}

static void
app_get_resource(const char *res_file_in, char *res_path_out, int res_path_max)
{
    char *res_path = app_get_resource_path();
    if (res_path) {
        snprintf(res_path_out, res_path_max, "%s%s", res_path, res_file_in);
        free(res_path);
    }
}
```

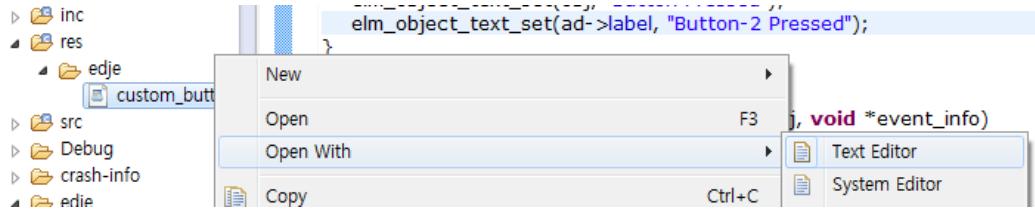
2번째 함수인 app_get_resource()에서 app_get_resource_path()를 호출했습니다. 이것은 /res 폴더의 절대 경로를 구해서 반환하는 함수입니다.

예제를 다시 실행하면 녹색 이미지가 적용된 5번째 Button 위젯을 볼 수 있습니다. Button을 클릭하면 빨간색 이미지로 변경됩니다. 클릭을 해제하면 다시 녹색 이미지로 되돌아가고, 캡션 텍스트가 변경됩니다.



EDJE 파일 전체를 설명하기에는 너무나 복잡해서 중요한것만 짚고 넘어가겠습니다. EDJE 파일을 에디터로 열어봅시다.

/res/edje/custom_button.edc 파일을 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [Open With > Text Editor]를 선택합니다.



파일 위쪽에는 define 구문이 보입니다. 이중에서 ICON_NORMAL은 Normal 상태의 Button 배경 이미지 파일명을 의미합니다. 만약 배경 이미지 파일명이 btn_n.png로 변경된다면 아래와 같이 수정하면 됩니다.

```
#define ICON_NORMAL btn_n.png
```

그 아래에 있는 ICON_PRESSED는 짐작하셨듯이 Pressed 상태의 Button 배경 이미지 파일명 입니다.

```
#define ICON_NORMAL green.png  
#define ICON_PRESSED red.png  
#define BUTTON_MIN_WIDTH 142  
#define BUTTON_MIN_HEIGHT 56  
#define BUTTON_PADDING_LEFT_RIGHT 8  
#define BUTTON_ICON_HEIGHT 46  
#define BUTTON_ICON_WIDTH 46  
#define BUTTON_TEXT_SIZE 30
```

좀더 아래로 내려가면 다음과 같은 코드가 있습니다.

```
collections {  
    base_scale: 1.8;  
    group { name: "elm/button/base/customized";  
        script {  
            public mouse_down = 0;  
            public multi_down = 0;  
        }  
    }  
}
```

group 하위의 name 속성은 테마명을 지정하는 항목입니다. 여기에 커스텀 테마명을 지정하고 소스코드에서 아래와 같이 적용하면 됩니다.

```
elm_object_style_set(btn, "customized");
```

6) 관련 API

Evas_Object *elm_button_add(Evas_Object *parent) : Button 위젯을 생성하는 API.

void evas_object_smart_callback_add(Evas_Object *obj, const char *event, Evas_Smart_Cb func, const void *data) : 컨테이너 혹은 위젯에 콜백 이벤트 함수를 지정하는 API. / 파라미터 : 이벤트가 발생하는 오브젝트, 이벤트 종류. "clicked"는 클릭 이벤트를 의미합니다., 콜백 함수명, 콜백 함수에 전달하는 데이터.

Eina_Bool elm_object_style_set(Evas_Object *obj, const char *style) : 위젯의 스타일을 지정하는 API. / 파라미터 : 스타일이 적용되는 위젯, 스타일 종류를 지정. 'icon_reorder' 는 재요청 아이콘 스타일, 'icon_expand_add' 는 플러스 아이콘 스타일, 'icon_expand_delete' 는 마이너스 아이콘 스타일, 'customized' 는 커스텀 스타일.

void elm_theme_extension_add(Elm_Theme *th, const char *item) : 테마 정보 파일을 등록하는 API. / 파라미터 : 테마, EDJE 파일의 경로.

char *app_get_resource_path(void) : /res 폴더의 절대 경로를 구해서 반환하는 API.

8. Bg 위젯으로 배경 만들기

위젯에 배경 컬러 또는 배경 이미지를 표시하는 방법은 2가지가 있습니다. 1번째는 EDJE를 사용해서 테마를 적용하는 것이고, 2번째는 Bg를 사용하는 방법이 있습니다. Bg 위젯을 사용하면 간편하게 배경을 표시할 수 있습니다.

1) 컬러 Bg 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 BgEx 으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 bgex.c 파일을 열고 create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void  
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)  
{  
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);  
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);  
    elm_table_pack(table, child, x, y, w, h);  
    evas_object_show(child);  
}
```

my_table_pack() 은 Table 컨테이너에 위젯 혹은 컨테이너를 추가하는 함수입니다. Table은 화면을 여러 개의 셀로 분할하여 원하는 영역에 위젯을 배치하는 컨테이너 입니다. Table을 사용하면 다양한 모니터 해상도를 지원할 수 있습니다.

my_table_pack() 함수의 파라미터는 순서대로 다음과 같습니다.

- Table 컨테이너
- 하위 윈도우
- 수평 셀 번호
- 수직 셀 번호
- 수평 셀 개수
- 수직 셀 개수

create_base_gui() 함수로 이동해서 Bg 위젯을 생성하겠습니다.

Conformant와 Label은 이번 예제에서 필요없어서 삭제 하겠습니다.

```
/* Conformant */
/*ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);*/

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "Hello EFL");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);
evas_object_show(ad->label);*/

/* Table */
Evas_Object *table = elm_table_add(ad->win);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
```

```

/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, table);
evas_object_show(table);

{
    /* Bg-1 Color */
    Evas_Object *bg = elm_bg_add(ad->win);
    my_table_pack(table, bg, 0, 0, 2, 2);
    elm_bg_color_set(bg, 66, 162, 206);
    evas_object_show(bg);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

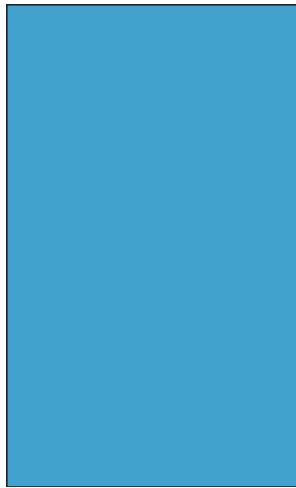
```

`Elm_table_add()` 는 Tabel 컨테이너를 생성하는 API 입니다.

`elm_bg_add()` 는 Bg 위젯을 생성하는 API 입니다.

`elm_bg_color_set()` 는 Bg의 배경 컬러를 지정하는 API 입니다. 1번째 파라미터에는 속성이 적용될 Bg 위젯을 지정합니다. 2번째부터 4번째 까지는 컬러값을 지정합니다. 순서대로 Red, Green, Blue 값을 0~255 범위로 입력하면 됩니다.

예제를 실행하면 화면이 파란색으로 변경되었습니다. 전체 영역에 Bg 위젯이 생성된 것입니다.

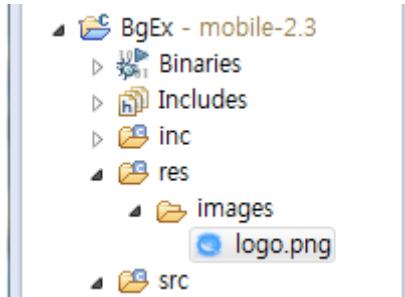


2) Bg 위젯에 이미지 적용 - 원본 사이즈

이번에는 배경 이미지가 적용된 Bg 위젯을 생성해 보겠습니다. 그러기 위해서 이미지 파일이 필요합니다.

배경 이미지로 사용할 이미지 파일을 복사하겠습니다. 소스 프로젝트 /res 폴더에 새로운 폴더를 생성하고 이름을 images라고 지정합니다. /res 폴더를 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [New > Folder]를 선택합니다. 팝업창이 나타나면 Folder name 항목에 images라고 입력하고 Finish 버튼을 누르면 새로운 폴더가 생성됩니다.

그런 다음 부록 /Image 폴더에 들어가서 logo.png 파일을 방금 생성한 /res/images 폴더로 복사합니다. 파일을 마우스 드래그하면 원하는 폴더로 복사할 수 있습니다.



새로운 Bg 위젯을 생성하고 배경 이미지를 지정해 보겠습니다.
create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
{
    /* Bg-1 Color */
    Evas_Object *bg = elm_bg_add(ad->win);
    my_table_pack(table, bg, 0, 0, 2, 2);
    elm_bg_color_set(bg, 66, 162, 206);
    evas_object_show(bg);

    /* Image path */
    char buf[PATH_MAX];
    app_get_resource("images/logo.png", buf, (int)PATH_MAX);

    /* Bg-2 Image Center */
    bg = elm_bg_add(ad->win);
    elm_bg_option_set(bg, ELM_BG_OPTION_CENTER);
    elm_bg_file_set(bg, buf, NULL);
    my_table_pack(table, bg, 2, 0, 2, 2);
}
```

app_get_resource() 는 /res 폴더의 절대경로를 구해서 하위 폴더 경로를
추가해서 반환해주는 함수입니다. 1번째 파라미터에는 하위 경로를
지정하는데 여기서는 'images/logo.png'를 전달했기 때문에 전체 경로는

~/res/images/logo.png 가 됩니다. 이 함수는 아직 생성되지 않았습니다.
잠시 후에 작성해 보도록 하겠습니다.

elm_bg_option_set() 은 이미지를 표시하는 스타일을 지정합니다. 2번째
파라미터에 ELM_BG_OPTION_CENTER를 지정하면 이미지를 원본 크기로
Bg 영역 중앙에 표시합니다.

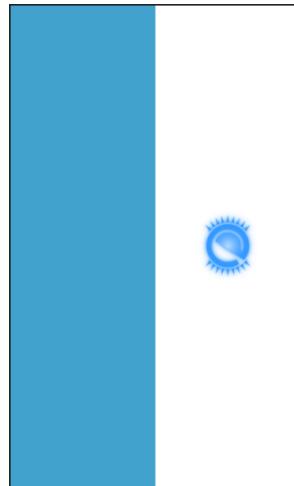
elm_bg_file_set() 는 Bg 위젯에 이미지 파일을 지정하는 함수입니다.
1번째 파라미터는 속성이 적용될 Bg 위젯을 지정하고, 2번째
파라미터에는 파일 경로를 전달하면 됩니다.

/res 폴더의 절대경로를 반환하는 함수를 생성하겠습니다.
create_base_gui() 함수 위에 새로운 코드를 추가합니다.

```
static void
app_get_resource(const char *res_file_in, char *res_path_out, int res_path_max)
{
    char *res_path = app_get_resource_path();
    if (res_path) {
        snprintf(res_path_out, res_path_max, "%s%s", res_path, res_file_in);
        free(res_path);
    }
}
```

소스코드는 ButtonEx 예제와 동일하기 때문에 자세한 설명은 생략하겠습니다.

예제를 다시 실행하면 화면 우측에 이미지가 표시됩니다. 우측이 2번째
Bg 영역이고 그 중앙에 이미지가 원본 사이즈로 출력된 것입니다.



3) Bg 위젯에 이미지 적용 - 원본 비율 유지해서 Resize

이번에는 원본 이미지의 가로/세로 비율을 유지한 상태로 영역에 채우는 방법을 알아보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

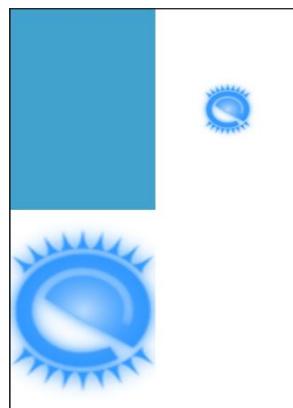
```
/* Bg-2 Image Center */
bg = elm_bg_add(ad->win);
elm_bg_option_set(bg, ELM_BG_OPTION_CENTER);
elm_bg_file_set(bg, buf, NULL);
my_table_pack(table, bg, 2, 0, 2, 2);

/* Bg-3 Image Scale */
bg = elm_bg_add(ad->win);
elm_bg_option_set(bg, ELM_BG_OPTION_SCALE);
elm_bg_file_set(bg, buf, NULL);
my_table_pack(table, bg, 0, 2, 2, 2);

}
```

`elm_bg_option_set()` 함수 2번째 파라미터에 `ELM_BG_OPTION_SCALE`을 지정하면 원본 이미지의 가로/세로 비율을 유지해서 Bg 영역에 채워줍니다.

예제를 다시 실행하면 화면 좌측 하단에 큰 이미지가 출력 됩니다.



4) Bg 위젯에 이미지 적용 - 원본 비율 무시해서 Resize

이번에는 원본 이미지의 가로/세로 비율을 무시하고 Bg 영역에 꽉 채우는 방법을 알아보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다. 4번째 Bg 위젯을 생성하는 코드입니다.

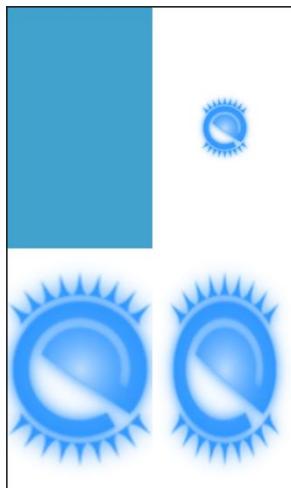
```
/* Bg-3 Image Scale */
bg = elm_bg_add(ad->win);
elm_bg_option_set(bg, ELM_BG_OPTION_SCALE);
elm_bg_file_set(bg, buf, NULL);
my_table_pack(table, bg, 0, 2, 2, 2);

/* Bg-4 Image Stretch */
bg = elm_bg_add(ad->win);
elm_bg_option_set(bg, ELM_BG_OPTION_STRETCH);
```

```
elm_bg_file_set(bg, buf, NULL);
my_table_pack(table, bg, 2, 2, 2, 2);
}
```

elm_bg_option_set() 함수 2번째 파라미터에 ELM_BG_OPTION_STRETCH를 지정하면 이미지를 Bg 영역에 꽉 채워줍니다.

예제를 다시 실행하면 화면 우측 하단에 큰 이미지가 출력 됩니다. 원본 이미지와 가로/세로 비율이 달라진 것을 알수 있습니다.



5) Bg 위젯에 이미지 적용 - Tile 스타일

이번에는 Tile 처럼 이미지를 반복해서 출력하는 방법을 알아보겠습니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다. 5번째 Bg 위젯을 생성하는 코드입니다.

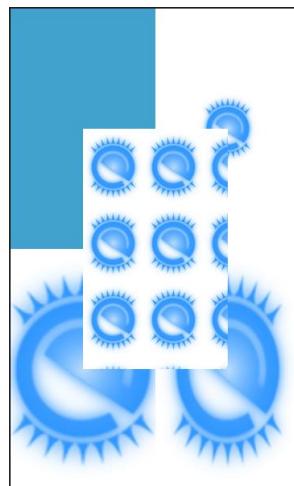
```
/* Bg-4 Image Stretch */
bg = elm_bg_add(ad->win);
elm_bg_option_set(bg, ELM_BG_OPTION_STRETCH);
elm_bg_file_set(bg, buf, NULL);
my_table_pack(table, bg, 2, 2, 2, 2);

/* Bg-5 Image Tile */
bg = elm_bg_add(ad->win);
elm_bg_option_set(bg, ELM_BG_OPTION_TILE);
elm_bg_file_set(bg, buf, NULL);
my_table_pack(table, bg, 1, 1, 2, 2);

}
```

`elm_bg_option_set()` 함수 2번째 파라미터에 `ELM_BG_OPTION_TILE` 을
지정하면 이미지를 반복해서 출력합니다.

예제를 다시 실행하면 화면 중앙에 이미지가 반복해서 출력 됩니다.



6) 관련 API

Evas_Object *elm_bg_add(Evas_Object *parent) : Bg 위젯을 생성하는 함수.

void elm_bg_color_set(Evas_Object *obj, int r, int g, int b) : Bg 위젯의 배경 컬러를 지정하는 함수. / 파라미터 : 속성이 적용될 Bg 위젯, 2번째부터 4번째 까지는 컬러값을 지정합니다. 순서대로 Red, Green, Blue 값을 0~255 범위로 입력하면 됩니다.

void elm_bg_option_set(Evas_Object *obj, Elm_Bg_Option option) : Bg 위젯에 이미지를 표시하는 스타일을 지정하는 함수. / 1번째 파라미터는 Bg 위젯 객체입니다. 2번째 파라미터는 이미지 배치 스타일입니다. 종류는 다음과 같습니다.

- ELM_BG_OPTION_CENTER : 이미지를 원본 크기로 Bg 영역 중앙에 표시.
- ELM_BG_OPTION_SCALE : 원본 이미지의 가로/세로 비율을 유지해서 Bg 영역에 채워줍니다.
- ELM_BG_OPTION_STRETCH : 이미지를 Bg 영역에 꽉 채워줍니다.
- ELM_BG_OPTION_TILE : 이미지를 반복해서 출력합니다.

Eina_Bool elm_bg_file_set(Evas_Object *obj, const char *file, const char *group) : Bg 위젯에 이미지 파일을 지정하는 함수. / 파라미터 : Bg 위젯 객체, 파일 경로.

9. Conformant 컨테이너로 화면 리사이즈

화면 위쪽 인디케이터(상태바)를 표시하려면 Conformant 컨테이너를 사용해야 합니다. Conformant가 있어도 인디케이터를 감출 수 있습니다. 키패드처럼 새로운 패널이 나타나서 화면 사이즈가 변경되어야 하는 경우에도 Conformant가 필요합니다. 예제를 통해서 Conformant 사용방법을 알아보겠습니다.

1) 인디케이터 감추기

새로운 소스 프로젝트를 생성하고 Project name을 ConformantEx 으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 bgex.c 파일을 열고 create_base_gui() 함수 위에 새로운 함수를 생성합니다. HelloWorld 예제에서 사용했던 함수입니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",  
     * "pad_large" and "pad_huge" available as styles in addition to the  
     * "default" frame style */  
    elm_object_style_set(frame, "pad_medium");  
    /* set the input weight/aling on the frame insted of the child */  
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
```

```

evas_object_size_hint_align_set(frame, h_align, v_align);
{
    /* tell the child that is packed into the frame to be able to expand */
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

    /* fill the expanded area (above) as opposed to center in it */
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    /* actually put the child in the frame and show it */
    evas_object_show(child);
    elm_object_content_set(frame, child);
}
/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}

```

인디케이터를 감추는 기능을 구현하기 위해서 Button을 추가해 봅시다.
create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다. Box
컨테이너를 생성하고 Button 위젯을 추가하는 코드입니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

```

```

/* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "Hello EFL");
    my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

    /* Button-1 */
    Evas_Object *btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "Hide");
    evas_object_smart_callback_add(btn, "clicked", btn_hide_cb, ad);
    my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
}
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

새로 추가된 Button을 클릭하면 인디케이터를 감추는 기능을 구현하겠습니다. create_base_gui() 함수 위에 Button 콜백 함수를 추가합니다.

```

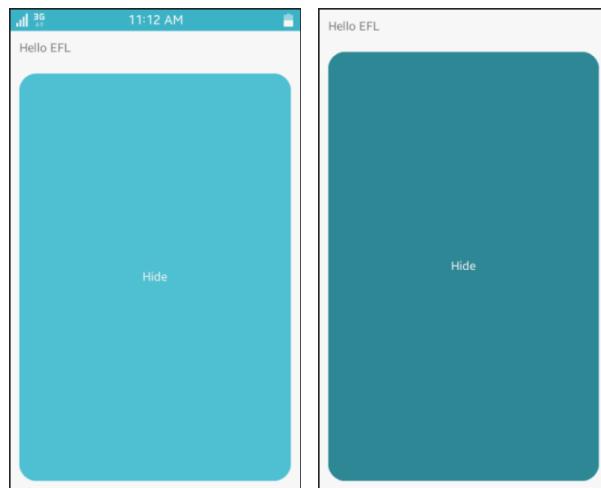
static void
btn_hide_cb(void *data, Evas_Object *obj, void *event_info)
{

```

```
    appdata_s *ad = (appdata_s*)data;  
    elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_HIDE);  
}  
_____
```

`elm_win_indicator_mode_set()` 는 인디케이터 모드를 변경하는 API입니다. 1번째 파라미터에는 Window 객체를 전달하면 됩니다. 앱은 오직 하나의 Window 만을 가질 수 있습니다. 2번째 파라미터에는 모드 종류를 지정합니다. `ELM_WIN_INDICATOR_HIDE`를 전달하면 인디케이터가 사라집니다.

예제를 빌드하고 실행하면 화면 위쪽에 인디케이터가 보입니다. Hide Button을 클릭하면 인디케이터가 사라집니다.



인디케이터가 있을 때와 없을 때 Button 위젯의 크기에 차이가 있습니다. Label 위젯의 높이를 최소로 지정했기 때문에 인디케이터가 사라졌을 때 Button이 그 만큼 높이가 더 길어지는 것입니다.

2) 인디케이터 보여주기

사라졌던 인디케이터를 다시 보여주는 기능을 구현하겠습니다.

create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다. 2번째 Button을 생성하는 코드입니다.

```
/* Button-1 */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Hide");
evas_object_smart_callback_add(btn, "clicked", btn_hide_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Show");
evas_object_smart_callback_add(btn, "clicked", btn_show_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
```

그런 다음 create_base_gui() 함수 위에 2번째 Button의 콜백 함수를 추가해 줍니다.

```
static void
btn_show_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = (appdata_s*)data;
    elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
}
```

예제를 다시 실행하고 Hide 버튼을 눌러봅시다. 인디케이터가 사라집니다.

이번에는 Show 버튼을 눌러봅시다. 인디케이터가 다시 나타납니다.



3) Entry 위젯 생성

Entry는 사용자로부터 텍스트를 입력받을 수 있는 에디터 위젯입니다. 키패드를 사용해서 텍스트를 입력하기 때문에 화면의 크기가 변경되어야 합니다. 방법은 Conformant 위에 Box 혹은 Layout 같은 다른 컨테이너를 올리고 그 위에 Entry를 추가하는 것입니다.

create_base_gui() 함수로 이동해서 끝 부분에 새로운 코드를 추가합니다. Entry 위젯을 추가하는 코드입니다.

```
/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Show");
evas_object_smart_callback_add(btn, "clicked", btn_show_cb, ad);
```

```

my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Entry */
Evas_Object *entry = elm_entry_add(ad->conform);
elm_object_text_set(entry, "Entry");
my_box_pack(box, entry, 1.0, 1.0, -1.0, -1.0);

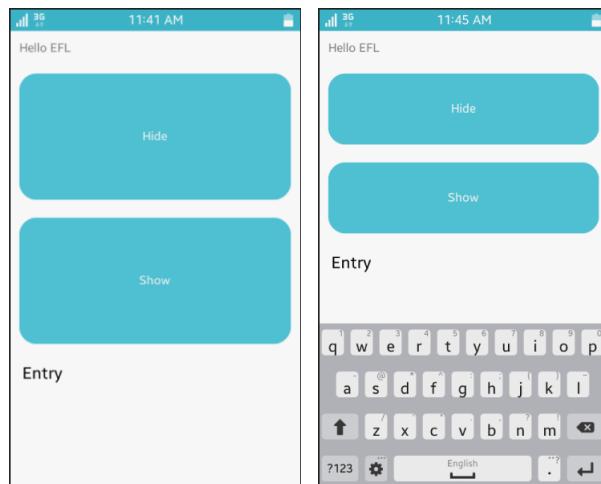
}

```

elm_entry_add() 는 Entry 위젯을 생성하는 API입니다. Conformant 위에 Entry를 생성한 것입니다.

예제를 빌드하고 실행시켜 봅시다. 화면 위쪽에는 Label 위젯이 보이고, 아래쪽에 Entry라는 텍스트가 Entry입니다. Entry의 배경 컬러가 화면 배경과 같아서 경계선이 모호합니다. 배경 컬러를 지정하는 방법은 다음 시간에 알아보겠습니다.

Entry라는 글자를 터치해 봅시다. 키패드가 나타나면서 화면이 수직 방향으로 줄어들었습니다. 덕분에 Entry가 제대로 보입니다.



4) 관련 API

Evas_Object *elm_entry_add(Evas_Object *parent) : Entry 위젯을 생성하는 API.

void elm_win_indicator_mode_set(Evas_Object *obj,
Elm_Win_Indicator_Mode mode) : 인디케이터 모드를 변경하는 API. /
파라미터 : Window 객체, 모드 종류. ELM_WIN_INDICATOR_HIDE를
전달하면 인디케이터가 사라집니다.

10. Entry 위젯 사용방법

사용자로부터 텍스트 문자열을 입력 받으려면 Entry 위젯을 사용하면 됩니다. Entry 위젯을 생성하고, 사용자가 입력한 내용을 구하는 방법을 알아보겠습니다.

1) Entry 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 EntryEx 으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 entryex.c 파일을 열고 create_base_gui() 함수 위에 새로운 함수를 생성합니다. Table 컨테이너에 위젯을 추가하는 함수입니다.

```
static void
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)
{
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
    elm_table_pack(table, child, x, y, w, h);
    evas_object_show(child);
}
```

그런 다음 create_base_gui() 함수로 이동해서 Box, Table 컨테이너와 Entry 위젯을 생성하겠습니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* Box to put the table in so we can bottom-align the table
 * window will stretch all resize object content to win size */
Evas_Object *box = elm_box_add(ad->conform);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

/* Table */
Evas_Object *table = elm_table_add(ad->conform);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 20 * elm_config_scale_get(), 20 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fill width but align to bottom of box */
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_box_pack_end(box, table);
evas_object_show(table);

}

```

```

/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
my_table_pack(table, ad->label, 0, 0, 4, 1);

/* Entry-1 */
Evas_Object *entry = elm_entry_add(ad->conform);
elm_entry_single_line_set(entry, EINA_TRUE);
elm_entry_entry_insert(entry, "Entry-1");
my_table_pack(table, entry, 0, 2, 4, 1);

}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Table은 화면 비율 단위로 위젯을 배치할 수 있는 컨테이너입니다. Entry의 배경으로 Bg를 사용하려면 같은 공간에 위치해야 하기 때문에 Table을 사용하는 것입니다. Box는 위젯 사이의 간격을 지정하기 위해서 사용하였습니다.

`elm_table_padding_set()` 는 안쪽 여백을 지정하는 API입니다.

`elm_entry_add()` 는 Entry 위젯을 생성하는 API입니다.

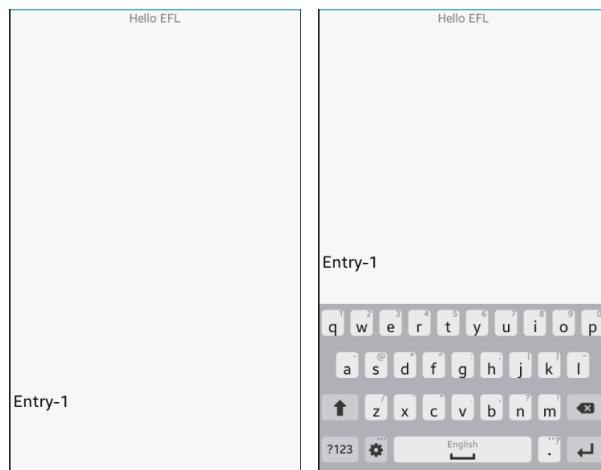
`elm_entry_single_line_set()` 는 멀티라인 지정&해제 API입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 한줄 입력 전용이 되고, EINA_FALSE를 전달하면 여러줄을 입력할 수 있습니다.

`elm_entry_entry_insert()` 는 Entry 위젯에 캡션 텍스트를 추가하는 API입니다. 기존에 입력되어 있는 텍스트의 뒤에 새로운 텍스트가 추가되는 것입니다.

소스 프로젝트를 빌드하고 실행시켜 봅시다. 화면 아래쪽에 Entry-1이라는 텍스트가 Entry 위젯입니다.

Entry 위젯을 터치하면 키패드가 나타나서 새로운 텍스트를 입력할 수 있습니다.

에뮬레이터에서는 키보드로 입력해도 됩니다.



2) Guide 텍스트 표시

에디터의 역할을 설명해 주는 텍스트를 Guide 텍스트 라고 합니다. 입력란이 비어있으면 Guide 텍스트가 나타나고, 키패드로 입력하면 Guide 텍스트는 사라집니다. 스마트폰은 화면이 작기 때문에 공간 활용을 효율적으로 해야합니다. Guide 텍스트를 사용하면 Label 위젯을 대신할 수 있어서 공간을 절약할 수 있습니다.

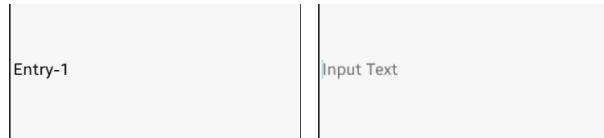
create_base_gui() 함수의 Entry 생성코드에 새로운 코드를 추가합니다.

```
/* Entry-1 */
```

```
Evas_Object *entry = elm_entry_add(ad->conform);
elm_entry_single_line_set(entry, EINA_TRUE);
elm_entry_entry_insert(entry, "Entry-1");
elm_object_part_text_set(entry, "elm.guide", "Input Text");
my_table_pack(table, entry, 0, 2, 4, 1);
```

elm_object_part_text_set() 함수의 2번째 파라미터에 'elm.guide'를 전달하면 Entry 위젯에 Guide 텍스트를 지정할 수 있습니다. Guide 텍스트 내용을 3번째 파라미터에 전달하면 됩니다.

예제를 다시 실행하고 Entry 위젯의 내용을 삭제해 봅시다. 'Input Text'라는 텍스트가 나타납니다. Entry 위젯에 텍스트를 입력하면 Guide 텍스트는 사라집니다.



3) Bg로 배경 만들기

Entry 위젯의 배경 컬러가 흰색이고, 화면 배경도 흰색이라서 Entry 위젯의 경계선이 확실하지 않습니다. Entry 위젯에 배경 컬러를 표시하는 방법은 2가지가 있습니다. 1번재는 Bg 위젯을 사용하는 것이고, 2번재는 EDJE를 사용하는 것입니다.

Bg를 배경으로 사용하는 방법을 알아보겠습니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다. Bg위젯을 생성하고 Entry 위젯과 동일한 영역좌표를 지정하는 코드입니다. Bg는 Entry 보다 먼저

생성되어야 한다는 점에 유의하기 바랍니다.

```
/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
my_table_pack(table, ad->label, 0, 0, 4, 1);

/* Bg-1 */
Evas_Object *bg = elm_bg_add(ad->conform);
elm_bg_color_set(bg, 170, 220, 255);
my_table_pack(table, bg, 0, 2, 4, 1);

/* Entry-1 */
Evas_Object *entry = elm_entry_add(ad->conform);
```

Bg 위젯을 Entry 보다 먼저 생성했고 영역좌표를 동일하게 지정하였습니다. 이렇게 하면 Bg가 Entry의 배경 처럼 보이게 됩니다.

다시 예제를 실행하면 Entry 위젯의 배경이 보입니다.



4) Entry에 입력된 내용 가져오기

Button을 누르면 사용자가 Entry에 입력한 텍스트를 구하는 방법을 알아보겠습니다. Entry 위젯을 이벤트 함수에서 사용하기 위해서 전역변수 혹은 AppData로 선언해야 합니다.

소스코드 제일 위쪽으로 이동하면 appdata_s 구조체가 정의되어 있습니다. 이것은 앱에서 사용하는 데이터를 저장하기 위한 구조체입니다. 기본적으로 Window, Conform, Label이 선언되어 있습니다. 여기에 Entry를 추가하겠습니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    Evas_Object *entry;  
} appdata_s;
```

그런 다음 create_base_gui() 함수로 되돌아 가서 끝부분에 새로운 코드를 추가합니다. Button 위젯을 생성하고 Entry 위젯을 AppData 구조체에 저장하는 코드입니다.

```
/* Label*/  
ad->label = elm_label_add(ad->conform);  
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");  
my_table_pack(table, ad->label, 0, 0, 4, 1);  
  
/* Button-1 */  
Evas_Object *btn = elm_button_add(ad->conform);  
elm_object_text_set(btn, "Get Text");  
evas_object_smart_callback_add(btn, "clicked", btn_clicked_cb, ad);  
my_table_pack(table, btn, 0, 1, 4, 1);  
  
/* Bg-1 */  
Evas_Object *bg = elm_bg_add(ad->conform);
```

```
elm_bg_color_set(bg, 170, 220, 255);
my_table_pack(table, bg, 0, 2, 4, 1);

/* Entry-1 */
Evas_Object *entry = elm_entry_add(ad->conform);
elm_entry_single_line_set(entry, EINA_TRUE);
elm_entry_entry_insert(entry, "Entry-1");
elm_object_part_text_set(entry, "elm.guide", "Input Text");
my_table_pack(table, entry, 0, 2, 4, 1);
ad->entry = entry;
```

Button 콜백 함수를 만들 차례입니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다. Entry 위젯의 텍스트를 구해서 Label 위젯에 표시하는 코드입니다.

```
static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s* ad = data;
    char* text = elm_entry_entry_get(ad->entry);
    elm_object_text_set(ad->label, text);
}
```

`elm_entry_entry_get()` 은 Entry 위젯의 텍스트를 구하는 함수입니다. `elm_entry_entry_insert()` 의 반대라고 생각하면 됩니다.

예제를 다시 실행하고 Entry 위젯에 텍스트를 변경한 다음, Button을 클릭해 봅시다. Entry에 입력한 텍스트가 Label에 표시됩니다.



5) 비밀번호 전용 Entry 위젯

비밀번호를 입력할 때는 주변 사람에게 보이지 않도록 텍스트를 와일드카드(*)로 표시해야 합니다. 비밀번호 입력 전용 Entry 위젯을 만들어 보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가하겠습니다. 2번째 Bg 위젯과 Entry 위젯을 생성하는 코드입니다.

```
/* Entry-1 */
Evas_Object *entry = elm_entry_add(ad->conform);
elm_entry_single_line_set(entry, EINA_TRUE);
elm_entry_entry_insert(entry, "Entry-1");
elm_object_part_text_set(entry, "elm.guide", "Input Text");
my_table_pack(table, entry, 0, 2, 4, 1);
ad->entry = entry;

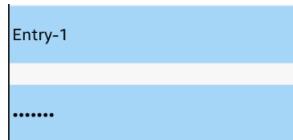
/* Bg-2 */
bg = elm_bg_add(ad->conform);
elm_bg_color_set(bg, 170, 220, 255);
my_table_pack(table, bg, 0, 3, 4, 1);

/* Entry-2 */
entry = elm_entry_add(ad->conform);
elm_entry_single_line_set(entry, EINA_TRUE);
elm_entry_entry_insert(entry, "Entry-2");
```

```
    elm_entry_password_set(entry, EINA_TRUE);
    my_table_pack(table, entry, 0, 3, 4, 1);
}
```

elm_entry_password_set() 은 비밀번호 전용 모드 지정 API입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 비밀번호 전용이되고, EINA_FALSE를 전달하면 해제됩니다.

예제를 다시 실행시켜 봅시다. 2번째 Entry에 내용을 입력하면 와일드카드로 표시되어 다른 사람이 알아볼수 없습니다.



6) Entry에 멀티라인 입력

Entry 위젯에 여러 줄의 텍스트를 입력하려면 Label 위젯과 동일한 방식을 사용하면 됩니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가하겠습니다. 3번째 Bg와 Entry를 추가하는 코드입니다.

```
/* Entry-2 */
entry = elm_entry_add(ad->conform);
elm_entry_single_line_set(entry, EINA_TRUE);
elm_entry_entry_insert(entry, "Entry-2");
elm_entry_password_set(entry, EINA_TRUE);
my_table_pack(table, entry, 0, 3, 4, 1);

/* Bg-3 */
bg = elm_bg_add(ad->conform);
```

```

elm_bg_color_set(bg, 170, 220, 255);
my_table_pack(table, bg, 0, 4, 4, 2);

/* Entry-3 */
entry = elm_entry_add(ad->conform);
elm_object_signal_emit(entry, "elm,state,scroll,enabled", "");
elm_object_text_set(entry, "<font_size=30><align=left>Once upon a time
there was a prince who was so selfish and unkind that he and all who lived in his
castle were put under a powerful spell.<br>The prince was turned into a terrible
beast.</align></font_size>");
my_table_pack(table, entry, 0, 4, 4, 2);
}

```

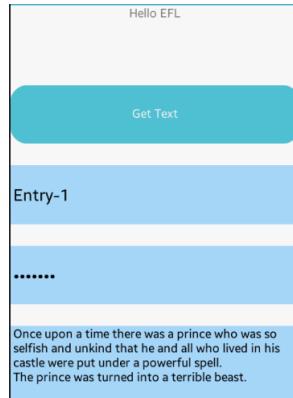
elm_object_text_set() 은 Label과 Button 위젯의 캡션 텍스트를 변경할 때 사용했던 API입니다. Entry에서도 사용 가능합니다. 다만 elm_object_signal_emit() 함수와 함께 사용해야 합니다. 2번째 파라미터에 html 태그를 전달하면 됩니다.

<font_size=20> 텍스트 폰트 크기를 지정하는 태그입니다.

<align=left> 는 수평 정렬방식을 왼쪽으로 지정하는 태그입니다.

 은 줄바꿈 태그입니다.

예제를 다시 실행시켜 봅시다. 3번째 Entry에 여러 줄의 텍스트가 출력되었습니다. 키패드를 이용해서 직접 텍스트를 입력해 봅시다. 폰트 크기가 다른 것을 알수 있습니다. html 태그로 지정한 속성은 출력에만 적용되고 입력에는 적용되지 않습니다.



7) 관련 API

Evas_Object *elm_entry_add(Evas_Object *parent) : Entry 위젯을 생성하는 API.

void elm_entry_single_line_set(Evas_Object *obj, Eina_Bool single_line) : 멀티라인 지정/해제 API. /
파라미터 : Entry 객체, 싱글라인 출력 여부. EINA_TRUE를 전달하면 한줄 입력 전용이 되고, EINA_FALSE를 전달하면 여러줄을 입력할 수 있습니다.

void elm_entry_entry_insert(Evas_Object *obj, const char *entry) : Entry 위젯에 캡션 텍스트를 추가하는 API. 기존에 입력되어 있는 텍스트의 뒤에 새로운 텍스트가 추가되는 것입니다.

void elm_object_part_text_set(Evas_Object *obj, const char *part, const char *text) : Entry 위젯에 가이드 텍스트 지정 API. / 파라미터 : Entry 객체, 텍스트 적용 부위. 'elm.guide'를 전달하면 Entry 위젯에 Guide 텍스트를 지정할 수 있습니다. Guide 텍스트 내용을 3번째 파라미터에 전달하면 됩니다.

char *elm_entry_entry_get(Evas_Object *obj) : Entry 위젯의 텍스트를
구하는 API. elm_entry_entry_insert() 의 반대라고 생각하면 됩니다.

void elm_entry_password_set(Evas_Object *obj, Eina_Bool
password) : 비밀번호 전용 모드 지정 API. 2번째 파라미터에
EINA_TRUE를 전달하면 비밀번호 전용이되고, EINA_FALSE를 전달하면
해제됩니다.

11. Check 위젯 사용방법

On/Off 중에서 선택을 하려면 Check 위젯을 사용하면 편리합니다.
Check 위젯을 생성하고, 사용자의 이벤트를 구하는 방법을
알아보겠습니다.

1) Check 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 CheckEx으로
지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 소스 파일(~.c)을 열고
appdata 구조체에 새로운 변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
Evas_Object *check1;
Evas_Object *check2;
Evas_Object *check3;
Evas_Object *check4;
} appdata_s;
```

4개의 Check 위젯 변수를 추가하였습니다. 소스코드로 Check 위젯을
생성해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를
추가합니다. Box 컨테이너에 위젯을 추가하는 함수입니다.

```

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}

```

그린 다음 `create_base_gui()` 함수로 이동해서 새로운 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

/* check 1 */
ad->check1 = elm_check_add(ad->conform);
elm_object_style_set(ad->check1,"popup");
elm_object_text_set(ad->check1, "Editable");
my_box_pack(box, ad->check1, 1.0, 1.0, -1.0, -1.0);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

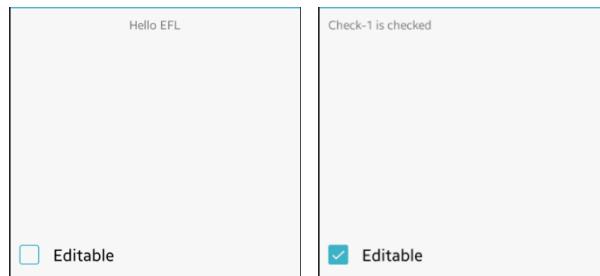
```

Box와 Check 위젯을 생성하였고, Box에 Label과 Check 위젯을 추가하였습니다.

elm_check_add(Evas_Object *parent) 는 Check 위젯을 생성하는 API입니다.

elm_object_text_set(obj, text) 는 위젯의 캡션 텍스트를 지정하는 API입니다.

예제를 빌드하고 실행시켜 보겠습니다. Check 위젯이 화면에 표시되었습니다. 사각형 부분을 터치하면 체크 마크가 나타납니다. 다시 한번 터치하면 체크 마크가 사라집니다.



2) Check 위젯 마크 변경하기

Check 위젯의 마크를 변경해 보겠습니다. 스타일을 변경하면 됩니다. create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* check 1 */
ad->check1 = elm_check_add(ad->conform);
elm_object_style_set(ad->check1,"popup");
elm_object_text_set(ad->check1, "Editable");
my_box_pack(box, ad->check1, 1.0, 1.0, -1.0, -1.0);
```

```

/* check 2 */
ad->check2 = elm_check_add(ad->conform);
elm_object_style_set(ad->check2, "favorite");
elm_object_text_set(ad->check2, "Favorite");
elm_check_state_set(ad->check2, EINA_TRUE);
my_box_pack(box, ad->check2, 1.0, 1.0, -1.0, -1.0);

/* check 3 */
ad->check3 = elm_check_add(ad->conform);
elm_object_style_set(ad->check3, "on&off");
elm_object_text_set(ad->check3, "On / Off");
elm_check_state_set(ad->check3, EINA_FALSE);
my_box_pack(box, ad->check3, 1.0, 1.0, -1.0, -1.0);

}

```

2번째와 3번째 Check 위젯을 생성하였습니다.

`elm_object_style_set(Evas_Object *obj, const char *style)` 는 객체의 스타일을 지정하는 API입니다. Check 위젯의 스타일 종류는 다음과 같습니다.

- favorite : 별마크
- on&off : 전원 On/Off 마크

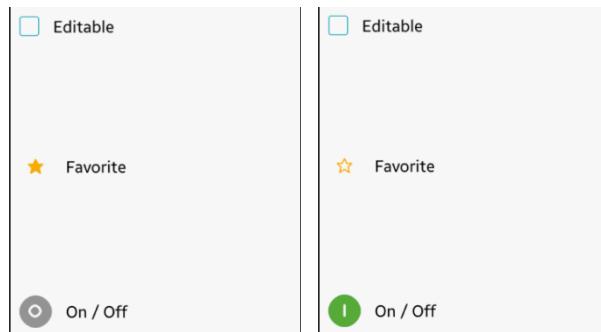
2번째 Check 위젯의 스타일을 "favorite" 으로 지정하였습니다. 별모양의 마크가 표시됩니다.

3번째 Check 위젯의 스타일을 "on&off" 으로 지정하였습니다. 전원 On/Off 마크가 표시됩니다.

`elm_check_state_set(Elm_Check *obj, Eina_Bool state)` 는 Check 위젯의 On/Off 상태를 지정하는 API입니다. 2번째 파라미터에 EINA_TRUE를

지정하면 On 상태가 되고, EINA_FALSE를 지정하면 Off 상태가 됩니다.

예제를 다시 실행시켜 봅시다. 2번째, 3번째 Check 위젯이 생성되었고 마크 모양이 변경되었습니다.



3) Check 위젯 On/Off 이벤트 구하기

사용자가 Check 위젯을 터치했을 때의 이벤트를 구해보겠습니다. 그러기 위해서 Check 위젯에 이벤트 콜백 함수를 지정합니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* check 1 */
ad->check1 = elm_check_add(ad->conform);
elm_object_style_set(ad->check1, "popup");
elm_object_text_set(ad->check1, "Editable");
elm_object_smart_callback_add(ad->check1, "changed", check_changed_cb, ad);
my_box_pack(box, ad->check1, 1.0, 1.0, -1.0, -1.0);

/* check 2 */
ad->check2 = elm_check_add(ad->conform);
elm_object_style_set(ad->check2, "favorite");
elm_object_text_set(ad->check2, "Favorite");
elm_check_state_set(ad->check2, EINA_TRUE);
```

```
evas_object_smart_callback_add(ad->check2, "changed", check_changed_cb, ad);
my_box_pack(box, ad->check2, 1.0, 1.0, -1.0, -1.0);

/* check 3 */
ad->check3 = elm_check_add(ad->conform);
elm_object_style_set(ad->check3, "on&off");
elm_object_text_set(ad->check3, "On / Off");
elm_check_state_set(ad->check3, EINA_FALSE);
evas_object_smart_callback_add(ad->check3, "changed", check_changed_cb, ad);
my_box_pack(box, ad->check3, 1.0, 1.0, -1.0, -1.0);
```

evas_object_smart_callback_add(Evas_Object *obj, char *event, Evas_Smart_Cb func, void *data) 는 컨테이너와 위젯 같은 스마트 객체에 콜백 함수를 지정하는 API입니다. 2번 파라미터에 "changed"를 전달하면 Check 위젯의 상태가 변경될 때 콜백 함수가 호출됩니다.

이제 콜백 함수를 생성할 차례입니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void
check_changed_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    int check_num = 0;
    if( obj == ad->check1 )
        check_num = 1;
    else if( obj == ad->check2 )
        check_num = 2;
    else if( obj == ad->check3 )
        check_num = 3;
    else
        return;
```

```
Eina_Bool state = elm_check_state_get(obj);
char buf[64];

sprintf(buf, "Check-%d is %s", check_num, state ? "checked" : "unchecked");
elm_object_text_set(ad->label, buf);

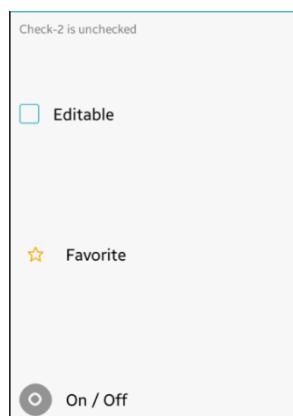
}
```

사용자가 몇 번째 Check 위젯을 터치했는지를 판단하고, 해당 위젯의 현재 상태를 구해서 Label 위젯에 상태를 표시하는 코드입니다.

Check 위젯 상태 변경 이벤트 함수의 1번째 파라미터에는 사용자 데이터가 전달되고, 2번째는 이벤트가 발생한 객체, 3번째는 이벤트 정보가 전달됩니다.

`elm_check_state_get(const Elm_Check *obj)` 는 Check 위젯의 On/Off 상태를 반환하는 API입니다. `elm_check_state_set()` 함수와 반대 역할입니다.

예제를 다시 실행시켜 봅시다. Check 위젯을 터치하면 선택된 위젯의 번호와 현재 상태가 Label 위젯에 표시됩니다.



4) Check 위젯 활성/비활성

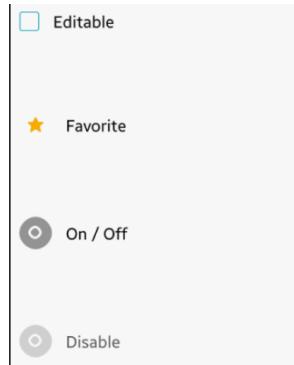
마지막으로 Check 위젯을 비활성 상태로 변경하는 방법을 알아보겠습니다. `create_base_gui()` 함수에 4번째 Check 위젯 생성 코드를 추가합니다.

```
/* check 3 */
ad->check3 = elm_check_add(ad->conform);
elm_object_style_set(ad->check3, "on&off");
elm_object_text_set(ad->check3, "On / Off");
elm_check_state_set(ad->check3, EINA_FALSE);
evas_object_smart_callback_add(ad->check3, "changed", check_changed_cb, ad);
my_box_pack(box, ad->check3, 1.0, 1.0, -1.0, -1.0);

/* check 4 */
ad->check4 = elm_check_add(ad->conform);
elm_object_style_set(ad->check4, "on&off");
elm_object_text_set(ad->check4, "Disable");
elm_object_disabled_set(ad->check4, EINA_TRUE);
evas_object_smart_callback_add(ad->check4, "changed", check_changed_cb,
ad);
my_box_pack(box, ad->check4, 1.0, 1.0, -1.0, -1.0);
}
```

`elm_object_disabled_set(Evas_Object *obj, Eina_Bool disabled)` 는 객체의 활성/비활성 상태를 변경하는 API입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 비활성 상태가 되고, EINA_FALSE를 전달하면 활성 상태가 됩니다.

다시 예제를 실행시켜 봅시다. 4번 Check 위젯이 추가되었습니다. 비활성 상태로 지정했기 때문에 터치해도 변화가 없습니다.



5) 관련 API

Evas_Object* elm_check_add(Evas_Object *parent) 는 Check 위젯을 생성하는 API입니다.

Eina_Bool elm_object_style_set(Evas_Object *obj, const char *style) 는 객체의 스타일을 지정하는 API입니다. Check 위젯의 스타일 종류는 다음과 같습니다.

- "popup" : 체크마크
- favorite : 별마크
- on&off : 전원 On/Off 마크

void elm_object_text_set(obj, text) 는 위젯의 캡션 텍스트를 지정하는 API입니다.

void elm_check_state_set(Elm_Check *obj, Eina_Bool state) 는 Check 위젯의 On/Off 상태를 지정하는 API입니다. 2번째 파라미터에 EINA_TRUE를 지정하면 On 상태가 되고, EINA_FALSE를 지정하면 Off 상태가 됩니다.

`void evas_object_smart_callback_add(Evas_Object *obj, char *event, Evas_Smart_Cb func, void *data)` 는 컨테이너와 위젯 같은 스마트 객체에 콜백 함수를 지정하는 API입니다. 2번 파라미터에 "changed"를 전달하면 Check 위젯의 상태가 변경될 때 콜백 함수가 호출됩니다.

`Eina_Bool elm_check_state_get(const Elm_Check *obj)` 는 Check 위젯의 On/Off 상태를 반환하는 API입니다. `elm_check_state_set()` 함수와 반대 역할입니다.

`void elm_object_disabled_set(Evas_Object *obj, Eina_Bool disabled)` 는 객체의 활성/비활성 상태를 변경하는 API입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 비활성 상태가 되고, EINA_FALSE를 전달하면 활성 상태가 됩니다.

12. Radio 위젯 사용방법

여러개의 메뉴 중에서 하나를 선택 하려면 Radio 위젯을 사용하면 편리합니다. Radio 위젯을 생성하고, 사용자의 이벤트를 구하는 방법을 알아보겠습니다.

1) Radio 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 RadioEx 으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 소스 파일(~.c)을 열고 create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box 컨테이너에 위젯을 추가하는 함수입니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",  
     * "pad_large" and "pad_huge" available as styles in addition to the  
     * "default" frame style */  
    elm_object_style_set(frame, "pad_medium");  
    /* set the input weight/aling on the frame insted of the child */  
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);  
    evas_object_size_hint_align_set(frame, h_align, v_align);  
    /* tell the child that is packed into the frame to be able to expand */
```

```

    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

    /* fill the expanded area (above) as opposed to center in it */
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    /* actually put the child in the frame and show it */
    evas_object_show(child);
    elm_object_content_set(frame, child);

}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);

}

```

그런 다음 `create_base_gui()` 함수로 이동해서 새로운 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

    /* child object - indent to how relationship */
    /* A box to put things in vertically - default mode for box */
    Evas_Object *box = elm_box_add(ad->win);
    evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

    elm_object_content_set(ad->conform, box);
    evas_object_show(box);
}
```

```

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "Select Radio");
    my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

    Evas_Object *radio, *radio_group;

    /* radio 1-1 */
    radio = elm_radio_add(ad->conform);
    elm_object_text_set(radio, "Cat");
    elm_radio_state_value_set(radio, 1);
    radio_group = radio;
    my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

    /* radio 1-2 */
    radio = elm_radio_add(ad->conform);
    elm_object_text_set(radio, "Dog");
    elm_radio_state_value_set(radio, 2);
    elm_radio_group_add(radio, radio_group);
    my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

    /* radio 1-3 */
    radio = elm_radio_add(ad->conform);
    elm_object_text_set(radio, "Hamster");
    elm_radio_state_value_set(radio, 3);
    elm_radio_group_add(radio, radio_group);
    my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

3개의 Radio를 생성하였습니다.

`elm_radio_add(Evas_Object *parent)` 는 Radio 위젯을 생성하는 API입니다.

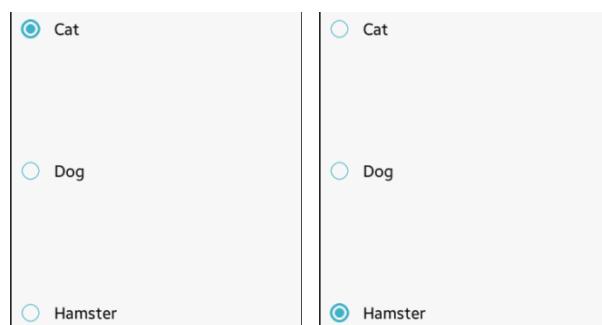
`elm_radio_state_value_set(Elm_Radio *obj, int value)` 는 Radio 위젯에 상태 값을 지정하는 API입니다. Radio 위젯은 여러개가 그룹으로 동작하기 때문에 각각의 위젯에 ID값을 서로 다르게 지정해야 합니다.

`radio_group` 는 Radio 그룹 변수인데 1번째 Radio 위젯을 Radio 그룹으로 사용하면 됩니다.

`elm_radio_group_add(Elm_Radio *obj, Evas_Object *group)` 는 Radio 그룹에 위젯을 추가하는 API입니다. 1번째 파라미터에는 추가되는 위젯을 전달하고, 2번째 파라미터에는 Radio 그룹을 전달하면 됩니다.

예제를 빌드하고 실행시켜 봅시다. 3개의 Radio 위젯이 화면에 나타나고 Radio 위젯을 터치하면 선택 마크가 표시됩니다.

다른 Radio 위젯을 선택하면 선택 마크의 위치가 이동합니다.



2) Radio 위젯 항목 선택 이벤트 구하기

사용자가 Radio 위젯을 터치했을 때의 이벤트를 구해서 몇 번째 항목이 선택되었는지를 확인해 보겠습니다. 그러기 위해서 Radio 위젯에 이벤트 콜백 함수를 지정합니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* radio 1-1 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Cat");
elm_radio_state_value_set(radio, 1);
radio_group = radio;
evas_object_smart_callback_add(radio, "changed", radio_animal_cb, ad);
my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

/* radio 1-2 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Dog");
elm_radio_state_value_set(radio, 2);
evas_object_smart_callback_add(radio, "changed", radio_animal_cb, ad);
elm_radio_group_add(radio, radio_group);
my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

/* radio 1-3 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Hamster");
elm_radio_state_value_set(radio, 3);
evas_object_smart_callback_add(radio, "changed", radio_animal_cb, ad);
elm_radio_group_add(radio, radio_group);
my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);
```

`evas_object_smart_callback_add(Evas_Object *obj, char *event, Evas_Smart_Cb func, void *data)` 는 컨테이너와 위젯 같은 스마트 객체에 콜백 함수를 지정하는 API입니다. 2번 파라미터에 "changed"를 전달하면 Radio 위젯의 상태가 변경될 때 콜백 함수가 호출됩니다.

이제 콜백 함수를 생성할 차례입니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void
radio_animal_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    int value = 0;
    value = elm_radio_value_get(obj);
    char buf[64];
    sprintf(buf, "Animal Radio : %d", value);

    // 1st Radio Group
    switch( value ) {
        case 1 :
            sprintf(buf, "%s %s ", buf, "Cat");
            break;
        case 2 :
            sprintf(buf, "%s %s ", buf, "Dog");
            break;
        case 3 :
            sprintf(buf, "%s %s ", buf, "Hamster");
            break;
    }
    elm_object_text_set(ad->label, buf);
}
```

사용자가 선택한 Radio 위젯의 상태값을 구해서 동물 이름과 함께 Label 위젯에 출력하는 코드입니다.

elm_radio_value_get(const Elm_Radio *obj) 는 Radio 위젯의 상태값을 반환하는 API입니다. 현재 선택된 Radio 위젯의 설정값을 반환합니다.

예제를 다시 실행시켜 봅시다. Radio 위젯을 터치하면 해당 위젯의 상태값과 동물 이름이 출력됩니다.



3) 2번째 Radio 그룹

Radio 위젯은 여러개의 위젯이 그룹으로 동작합니다. 3개의 Radio 위젯을 더 추가하고 2개의 Radio 그룹으로 분리하는 방법을 알아보겠습니다. create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* radio 1-3 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Hamster");
elm_radio_state_value_set(radio, 3);
evas_object_smart_callback_add(radio, "changed", radio_animal_cb, ad);
elm_radio_group_add(radio, radio_group);
```

```

my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

/* radio 2-1 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Cookie");
elm_radio_state_value_set(radio, 1);
radio_group = radio;
evas_object_smart_callback_add(radio, "changed", radio_dessert_cb, ad);
my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

/* radio 2-2 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Icecream");
elm_radio_state_value_set(radio, 2);
evas_object_smart_callback_add(radio, "changed", radio_dessert_cb, ad);
elm_radio_group_add(radio, radio_group);
my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

/* radio 2-3 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Juice");
elm_radio_state_value_set(radio, 3);
evas_object_smart_callback_add(radio, "changed", radio_dessert_cb, ad);
elm_radio_group_add(radio, radio_group);
my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

}

```

3개의 Radio 위젯을 추가하는 코드입니다. `elm_radio_state_value_set()` 함수를 사용해서 상태값을 각각 1, 2, 3 으로 지정했습니다.

1번째 Radio 위젯을 Radio 그룹으로 지정했으며, `elm_radio_group_add()` 함수를 사용해서 2번째, 3번째 Radio 위젯을 Radio 그룹에 추가하였습니다.

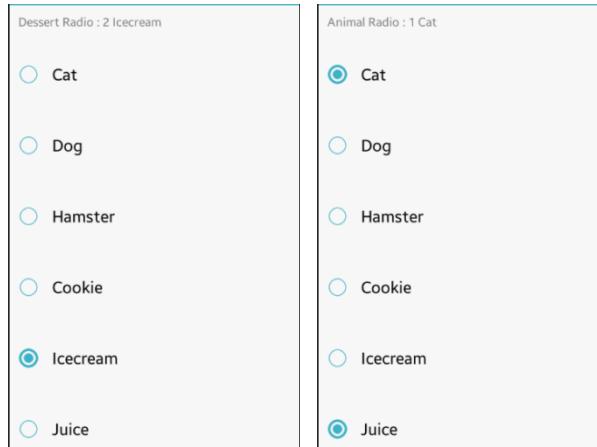
그리고 이벤트 콜백 함수명을 `radio_dessert_cb` 으로 지정했습니다. 이제 콜백 함수를 생성할 차례입니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void
radio_dessert_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    int value = 0;
    value = elm_radio_value_get(obj);
    char buf[64];
    sprintf(buf, "Dessert Radio : %d", value);

    switch( value ) {
        case 1 :
            sprintf(buf, "%s %s ", buf, "Cookie");
            break;
        case 2 :
            sprintf(buf, "%s %s ", buf, "Icecream");
            break;
        case 3 :
            sprintf(buf, "%s %s ", buf, "Juice");
            break;
    }
    elm_object_text_set(ad->label, buf);
}
```

함수의 내용은 `radio_animal_cb()` 함수와 거의 유사합니다.

예제를 다시 실행시키면 총 6개의 Radio 위젯이 나타납니다. Radio 위젯을 터치하면 1번째 그룹과 2번째 그룹이 별도로 마크가 표시됩니다.



4) 소스코드로 Radio 선택 항목 변경

예제가 실행되면 자동으로 2번째 그룹의 1번째 항목이 선택되는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```

/* radio 2-3 */
radio = elm_radio_add(ad->conform);
elm_object_text_set(radio, "Juice");
elm_radio_state_value_set(radio, 3);
evas_object_smart_callback_add(radio, "changed", radio_dessert_cb, ad);
elm_radio_group_add(radio, radio_group);
my_box_pack(box, radio, 1.0, 1.0, -1.0, -1.0);

/* Set selection to 2nd radio */
elm_radio_value_set(radio_group, 1);

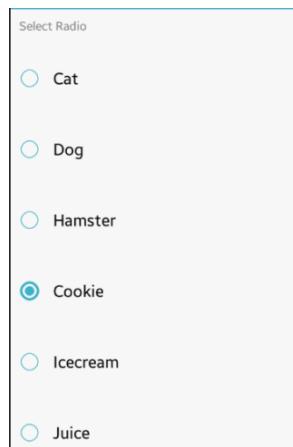
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

`elm_radio_value_set(Elm_Radio *obj, int value)` 는 Radio 그룹의 설정값을 지정하는 API입니다. 설정값이 일치하는 Radio 위젯이 선택 설정됩니다.

예제를 다시 실행시켜 봅시다. 2번째 그룹의 1번째 항목이 자동으로 선택되었습니다.



5) 관련 API

`Evas_Object* elm_radio_add(Evas_Object *parent)` : Radio 위젯을 생성하는 API.

`void elm_radio_state_value_set(Elm_Radio *obj, int value)` : Radio 위젯에 상태 값을 지정하는 API. Radio 위젯은 여러개가 그룹으로 동작하기 때문에 각각의 위젯에 ID값을 서로 다르게 지정해야 합니다.

`void elm_radio_group_add(Elm_Radio *obj, Evas_Object *group)` : Radio 그룹에 위젯을 추가하는 API. 1번째 파라미터에는 추가되는 위젯을 전달하고, 2번째 파라미터에는 Radio 그룹을 전달하면 됩니다.

`void evas_object_smart_callback_add(Evas_Object *obj, char *event, Evas_Smart_Cb func, void *data)` : 컨테이너와 위젯 같은 스마트 객체에 콜백 함수를 지정하는 API. 2번 파라미터에 "changed"를 전달하면 Radio 위젯의 상태가 변경될 때 콜백 함수가 호출됩니다.

`int elm_radio_value_get(const Elm_Radio *obj)` : Radio 위젯의 상태값을 반환하는 API. 현재 선택된 Radio 위젯의 설정값을 반환합니다.

`void elm_radio_value_set(Elm_Radio *obj, int value)` : Radio 그룹의 설정값을 지정하는 API. 설정값이 일치하는 Radio 위젯이 선택 설정됩니다.

13. Popup 사용방법

사용자에게 간단한 메시지를 표시할 때 Popup을 사용하면 편리합니다.
일정시간이 경과한 후에 자동으로 Popup 이 닫히도록 할 수 있으며
사용자의 입력을 받을 수도 있습니다.

1) Button 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 PopupEx 으로
지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 popupex.c 파일을 열고
appdata_s 구조체에 새로운 변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    Evas_Object *box;
    Evas_Object *popup;
    Evas_Object *entry;
    int popupNum;
} appdata_s;
```

box는 위젯들을 순차적으로 배치하기 위한 컨테이너 입니다.

popup은 팝업 위젯의 핸들입니다. 팝업을 닫거나 데이터를 전달할 때
사용하게 됩니다.

entry는 팝업창에서 사용자가 텍스트를 입력할 때 사용됩니다.

popupNum는 현재 팝업의 인덱스 번호를 저장하게 됩니다.

create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box 컨테이너에 위젯을 추가하는 함수입니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
}
```

```
/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}
```

그런 다음 `create_base_gui()` 함수로 이동해서 4개의 Button 위젯을 생성하겠습니다. 이번 예제에서는 4가지 종류의 Popup을 만들어 볼 것입니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

    /* child object - indent to how relationship */
    /* A box to put things in vertically - default mode for box */
    ad->box = elm_box_add(ad->win);
    evas_object_size_hint_weight_set(ad->box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    elm_object_content_set(ad->conform, ad->box);
    evas_object_show(ad->box);

    {

        /* Label*/
        ad->label = elm_label_add(ad->conform);
```

```

elm_object_text_set(ad->label, "Please click a button below");
my_box_pack(ad->box, ad->label, 1.0, 0.0, 0.5, 0.0);

/* Button-1 */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Popup Text");
evas_object_smart_callback_add(btn, "clicked", make_popup_text, ad);
my_box_pack(ad->box, btn, 1.0, 0.0, -1.0, -1.0);

/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Popup 1 Button");
evas_object_smart_callback_add(btn, "clicked",
make_popup_text_1button, ad);
my_box_pack(ad->box, btn, 1.0, 0.0, -1.0, -1.0);

/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Popup 3 Buttons");
evas_object_smart_callback_add(btn, "clicked",
make_popup_text_3button, ad);
my_box_pack(ad->box, btn, 1.0, 0.0, -1.0, -1.0);

/* Button-4 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Popup Input Text");
evas_object_smart_callback_add(btn, "clicked", make_popup_input_text,
ad);
/* Note: this last button has weight 1 and align 0 so that the whole UI
is
* nicely and tightly packed at the top of the window.
*/
my_box_pack(ad->box, btn, 1.0, 1.0, -1.0, 0.0);
}

}

```

```
/* Show window after base gui is set up */  
evas_object_show(ad->win);
```

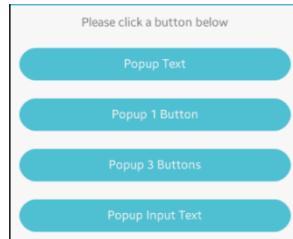
Conformant 위에 Box를 생성하였고, Box 위에 1개의 Label과 4개의 Button 위젯을 추가하였습니다.

Button의 콜백 함수를 생성해 주어야 빌드 오류가 발생하지 않습니다. `create_base_gui()` 함수 위에 4개의 새로운 함수를 추가합니다. 함수 내용은 잠시 후에 정의하겠습니다.

```
static void  
make_popup_text(void *data, Evas_Object *obj, void *event_info)  
{ }  
  
static void  
make_popup_text_1button(void *data, Evas_Object *obj, void *event_info)  
{ }  
  
static void  
make_popup_text_3button(void *data, Evas_Object *obj, void *event_info)  
{ }  
  
static void  
make_popup_input_text(void *data, Evas_Object *obj, void *event_info)  
{ }
```

소스 프로젝트를 빌드하고 실행하면 화면에 Label 과 Button이 보입니다. 1번째부터 3번째 Button은 차지하는 영역의 높이를 최소로 지정하였습니다. 마지막 4번째 Button은 `my_box_pack()` 함수의 4번째

파라미터에 1.0을 전달하여 영역 높이를 최대로 지정하였습니다. 그리고 6번째 파라미터에 0.0을 전달하여 Button의 높이를 최소로 지정하였습니다. 이렇게 하면 위젯들이 위쪽으로 몰리게 됩니다.



2) 텍스트 Popup 생성

텍스트 메시지를 표시하는 가장 기본적인 Popup을 생성해 보겠습니다. 1번째 Button의 콜백 함수에 Popup을 생성하는 코드를 추가합니다.

```
static void  
make_popup_text(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    ad->popup = elm_popup_add(ad->grid);  
    elm_popup_align_set(ad->popup, ELM_NOTIFY_ALIGN_FILL, 1.0);  
    evas_object_size_hint_weight_set(ad->popup, EVAS_HINT_EXPAND,  
                                     EVAS_HINT_EXPAND);  
    elm_object_text_set(ad->popup, "Text popup - timeout of 3 sec is set.");  
  
    evas_object_show(ad->popup);  
    ad->popupNum = 1;  
}
```

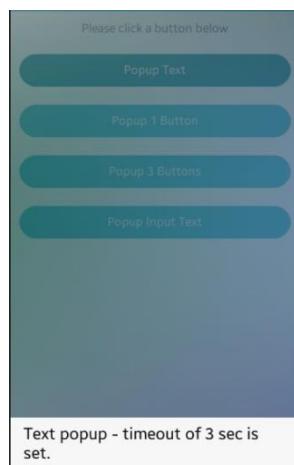
elm_popup_add() 는 Popup을 생성하는 API입니다.

`elm_popup_align_set()` 는 Popup의 위치를 지정하는 API입니다. 2번째 파라미터는 수평 위치를 지정하며, `ELM_NOTIFY_ALIGN_FILL`은 화면 수평 영역 전체를 사용하는 옵션입니다. 3번째 파라미터는 수직 위치를 지정하며 수치 범위는 0.0~1.0 사이입니다.

`evas_object_size_hint_weight_set()` 은 위젯의 크기에 대한 힌트를 지정하는 API입니다. 2번째 파라미터는 넓이이고, 3번째 파라미터는 높이를 의미합니다. `EVAS_HINT_EXPAND`는 주어진 영역에 가능한 크게 지정하는 옵션입니다.

popupNum에 1을 저장한 것은 1번째 Popup이라는 것을 기억하기 위해서 입니다.

예제를 다시 실행하고 1번째 Button을 클릭해 봅시다. 화면 아래쪽에 Popup이 나타나고 텍스트 메시지가 표시됩니다.



3) 자동 종료 타이머 지정

Popup에 버튼이 없으니 닫을수가 없습니다. 일정한 시간이 경과하면 자동으로 Popup이 닫히는 기능을 구현해 복시다.

make_popup_text() 함수에 새로운 코드를 추가합니다.

```
~  
elm_object_text_set(ad->popup, "Text popup - timeout of 3 sec is set.");  
elm_popup_timeout_set(ad->popup, 3.0);  
evas_object_smart_callback_add(ad->popup, "timeout", popup_timeout, ad);  
evas_object_show(ad->popup);  
~
```

elm_popup_timeout_set() 는 Popup에 타이머 이벤트를 지정하는 API입니다. 2번째 파라미터는 시간 간격을 지정합니다. 3.0을 지정하면 3초 후에 타이머 이벤트가 발생합니다.

evas_object_smart_callback_add() 는 이벤트를 수신하는 콜백 함수를 지정하는 API입니다. 2번째 파라미터에 'timeout'을 지정하면 타이머 이벤트가 발생할 때 콜백 함수가 호출됩니다. 3번째 파라미터에는 콜백 함수 이름을 지정합니다.

이제 타이머 이벤트 함수를 생성할 차례입니다. make_popup_text() 함수 위에 새로운 함수를 추가합니다.

함수 내용은 Popup을 삭제하고 Label 위젯에 텍스트를 출력하는 코드입니다.

```
static void  
popup_timeout(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    evas_object_del(obj);  
    elm_object_text_set(ad->label, "Time out");  
}
```

evas_object_del() 는 객체를 삭제하는 API입니다. 여기서는 타이머 이벤트 주체를 삭제했습니다. 바로 Popup을 삭제하는 것입니다.

다시 예제를 실행하고 1번째 버튼을 눌러봅시다. Popup이 나타났다가 잠시 후에 자동으로 사라집니다.

4) Block 영역 Touch하면 Popup 닫기

Popup이 나타난 상태에서 Popup 이외의 영역을 클릭하면 Popup이 닫히는 기능을 구현해 보겠습니다.

make_popup_text() 함수 끝부분에 새로운 코드를 추가합니다.

```
    evas_object_smart_callback_add(ad->popup, "timeout", popup_timeout, ad);  
    evas_object_smart_callback_add(ad->popup, "block_clicked", popup_block_clicked,  
ad);  
    evas_object_show(ad->popup);
```

evas_object_smart_callback_add() 함수의 1번째 파라미터에 Popup을 전달했고, 2번째 파라미터에 'block_clicked'를 전달했습니다. 이렇게 하면

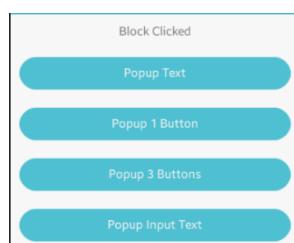
Popup 이외의 영역을 클릭했을 때 이벤트를 구할 수 있습니다.

이 이벤트를 구하는 함수를 생성하겠습니다. `make_popup_text()` 함수 위에 새로운 함수를 추가합니다.

함수 내용은 Popup을 삭제하고 Label 위젯에 텍스트를 출력하는 코드입니다.

```
static void  
popup_block_clicked(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    evas_object_del(obj);  
    elm_object_text_set(ad->label, "Block Clicked");  
}
```

다시 예제를 실행하고 1번째 버튼을 눌렀다가 Popup 이 나타나면 그 외 영역을 클릭해 봅시다. Popup이 사라지고 Label 위젯에 'Block Clicked'라는 텍스트가 표시됩니다.



5) Popup에 Button 추가하기

2번째 Button을 누르면 Popup 위에 Button이 생성되고, 그 Button을 클릭하면 Popup이 사라지는 기능을 구현해 보겠습니다.

make_popup_text_1button() 함수에 코드를 추가합니다. Popup을 생성하고, 그 위에 Button을 생성하는 코드입니다.

```
static void  
make_popup_text_1button(void *data, Evas_Object *obj, void *event_info)  
{  
    Evas_Object *btn;  
    appdata_s *ad = data;  
  
    /* popup */  
    ad->popup = elm_popup_add(ad->grid);  
    elm_popup_align_set(ad->popup, ELM_NOTIFY_ALIGN_FILL, 1.0);  
    evas_object_smart_callback_add(ad->popup, "block,clicked", popup_block_clicked,  
ad);  
    evas_object_size_hint_weight_set(ad->popup, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
    elm_object_text_set(ad->popup, "1Button popup");  
  
    /* ok button */  
    btn = elm_button_add(ad->popup);  
    elm_object_text_set(btn, "OK");  
    elm_object_part_content_set(ad->popup, "button1", btn);  
    evas_object_smart_callback_add(btn, "clicked", popup_btn1_clicked, ad);  
  
    evas_object_show(ad->popup);  
    ad->popupNum = 2;  
}
```

Popup 위에 추가된 Button을 클릭했을 때 Label에 텍스트를 출력하는 기능을 구현해 봅시다. make_popup_text_1button() 함수 위에 새로운 함수를 추가합니다. OK Button 이벤트 함수입니다.

```
static void
popup_btn1_clicked(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    const char *input;
    Eina_Strbuf *str;

    /* use eina_strbuf here for safe string allocation and formatting */
    input = elm_entry_entry_get(ad->entry);
    str = eina_strbuf_new();
    eina_strbuf_append_printf(str, "Input: '%s'", input);
    elm_object_text_set(ad->label, eina_strbuf_string_get(str));
    eina_strbuf_free(str);

    /* Destroy the popup AFTER reading from its child entry */
    evas_object_del(ad->popup);
    ad->popup = NULL;
}
```

eina_strbuf_new() 는 Strbuf 객체를 생성하는 API입니다. StrBuf 는 문자열을 쉽게 사용할수 있는 구조체입니다.

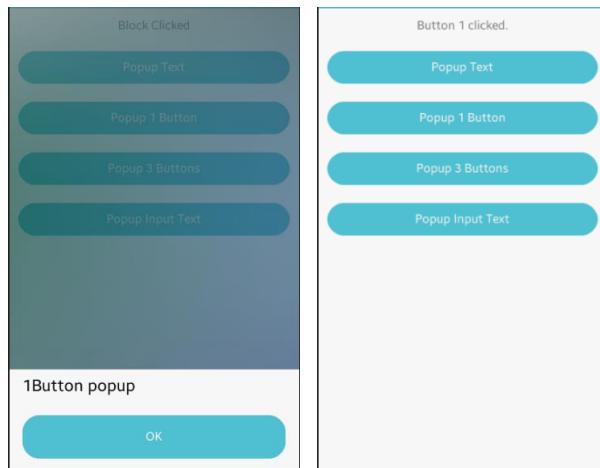
eina_strbuf_append_printf() 는 Strbuf에 새로운 문자열을 추가하는 API입니다.

eina_strbuf_string_get() 는 Strbuf에 저장된 문자열을 반환하는 API입니다.

eina_strbuf_free() 는 Strbuf 객체를 삭제하는 API입니다.

예제를 다시 실행하고 2번째 Button을 클릭해 봅시다. Popup이 나타나고 텍스트 메시지와 Button이 보입니다.

OK Button을 클릭하면 Popup이 사라지고 Label 위젯에 텍스트가 변경됩니다.



6) Popup에 Button 3개 추가하기

이번에는 Popup에 3개의 Button을 추가해 보겠습니다.

make_popup_text_3button() 함수에 내용을 추가합니다. Popup을 생성하고 그 위에 3개의 Button을 생성하는 코드입니다.

make_popup_text_1button() 함수의 내용을 복사해서 수정하면 타이핑 시간이 절약될 것입니다.

```
static void  
make_popup_text_3button(void *data, Evas_Object *obj, void *event_info)
```

```

{
  Evas_Object *btn;
  appdata_s *ad = data;

  /* popup */
  ad->popup = elm_popup_add(ad->grid);
  elm_popup_align_set(ad->popup, ELM_NOTIFY_ALIGN_FILL, 1.0);
  evas_object_smart_callback_add(ad->popup, "block,clicked", popup_block_clicked,
ad);
  evas_object_size_hint_weight_set(ad->popup, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
  elm_object_text_set(ad->popup, "3Button popup");

  /* ok button */
  btn = elm_button_add(ad->popup);
  elm_object_text_set(btn, "OK");
  elm_object_part_content_set(ad->popup, "button1", btn);
  evas_object_smart_callback_add(btn, "clicked", popup_btn1_clicked, ad);

  /* cancel button */
  btn = elm_button_add(ad->popup);
  elm_object_text_set(btn, "Cancel");
  elm_object_part_content_set(ad->popup, "button2", btn);
  evas_object_smart_callback_add(btn, "clicked", popup_btn2_clicked, ad);

  /* close button */
  btn = elm_button_add(ad->popup);
  elm_object_text_set(btn, "Close");
  elm_object_part_content_set(ad->popup, "button3", btn);
  evas_object_smart_callback_add(btn, "clicked", popup_btn3_clicked, ad);

  evas_object_show(ad->popup);
  ad->popupNum = 3;
}

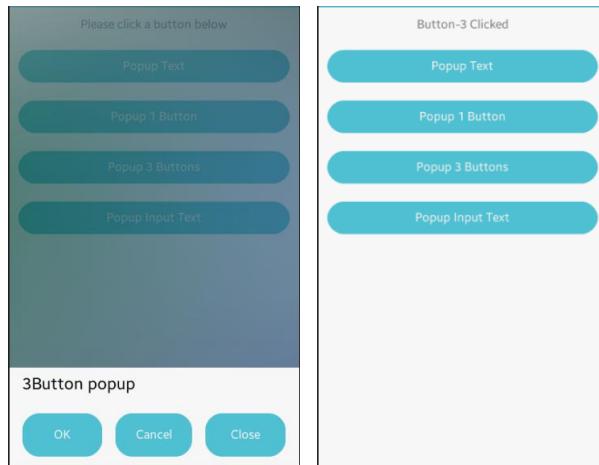
```

Popup 위에 있는 Button을 클릭했을 때 Label에 텍스트를 출력하는 기능을 구현해 봅시다. make_popup_text_3button() 함수 위에 새로운 함수 2개를 추가합니다. Cancel Button과 Close Button의 이벤트 함수입니다.

```
static void  
popup_btn2_clicked(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    evas_object_del(ad->popup);  
    elm_object_text_set(ad->label, "Button-2 Clicked");  
}  
  
static void  
popup_btn3_clicked(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    evas_object_del(ad->popup);  
    elm_object_text_set(ad->label, "Button-3 Clicked");  
}
```

예제를 다시 실행하고 3번째 Button을 누르면 Popup 위에 3개의 Button이 나타납니다.

각각의 Button을 누르면 Popup이 사라지고 Label에 텍스트가 변경됩니다.



7) Popup에 Entry 위젯 추가하기

이번에는 Popup에 Entry를 추가해서 사용자가 텍스트를 입력하는 기능을 구현해 보겠습니다. `make_popup_input_text()` 함수에 내용을 추가합니다. Popup을 생성하고 그 위에 1개의 Entry와 2개의 Button을 생성하는 코드입니다.

`make_popup_text_1button()` 함수의 내용을 복사해서 수정하면 타이핑 시간이 절약될 것입니다.

```

static void
make_popup_input_text(void *data, Evas_Object *obj, void *event_info)
{
    Evas_Object *btn;
    appdata_s *ad = data;
    Evas_Object *entry;

    /* popup */
    ad->popup = elm_popup_add(ad->grid);
    elm_popup_align_set(ad->popup, ELM_NOTIFY_ALIGN_FILL, 1.0);

```

```

    evas_object_smart_callback_add(ad->popup, "block_clicked", popup_block_clicked,
ad);
    evas_object_size_hint_weight_set(ad->popup, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    elm_object_part_text_set(ad->popup, "title,text", "Input Text");

/* entry */
entry = elm_entry_add(ad->popup);
evas_object_size_hint_weight_set(entry, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(entry, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_part_content_set(ad->popup, "elm.swallow.content" , entry);
evas_object_show(entry);
ad->entry = entry;

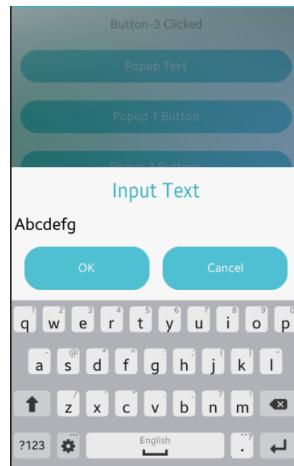
/* OK button */
btn = elm_button_add(ad->popup);
elm_object_text_set(btn, "OK");
elm_object_part_content_set(ad->popup, "button1", btn);
evas_object_smart_callback_add(btn, "clicked", popup_btn1_clicked, ad);

/* Cancel button */
btn = elm_button_add(ad->popup);
elm_object_text_set(btn, "Cancel");
elm_object_part_content_set(ad->popup, "button2", btn);
evas_object_smart_callback_add(btn, "clicked", popup_btn2_clicked, ad);

evas_object_show(ad->popup);
ad->popupNum = 4;
}

```

예제를 다시 실행하고 4번째 Button을 클릭하면 Popup에 Entry가 추가되어 있습니다.



사용자가 Entry에 텍스트를 입력하고 OK Button을 클릭하면 그 내용이 Label에 표시되는 기능을 구현해 봅시다.

popup_btn1_clicked() 함수에 새로운 코드를 추가합니다. 현재 Popup이 4번째라면 Entry의 캡션 텍스트를 구해서 Label에 표시하는 코드입니다. 그리고 Entry 위젯을 초기화 합니다.

```
static void
popup_btn1_clicked(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    if (ad->popupNum == 4) {
        const char *input;
        Eina_Strbuf *str;

        /* use eina_strbuf here for safe string allocation and formatting */
    }
}
```

```

input = elm_entry_entry_get(ad->entry);
str = eina_strbuf_new();
eina_strbuf_append_printf(str, "Input: '%s'", input);
elm_object_text_set(ad->label, eina_strbuf_string_get(str));
eina_strbuf_free(str);

} else {
    elm_object_text_set(ad->label, "Button 1 clicked.");
}

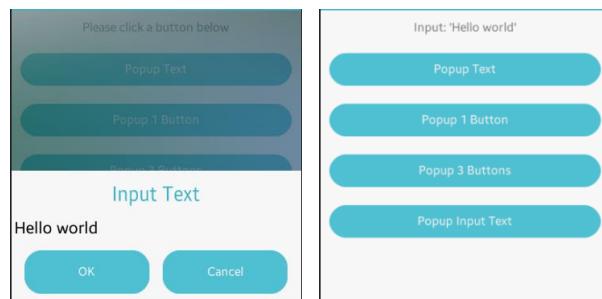
/* Destroy the popup AFTER reading from its child entry */
evas_object_del(ad->popup);
ad->popup = NULL;

/* Entry will be deleted when the popup is deleted (child widget) */
ad->entry = NULL;
}

```

예제를 다시 실행하고 4번째 Button을 클릭합니다. Popup이 나타나면 Entry에 텍스트를 입력하고 OK 버튼을 눌러봅시다.

Popup이 사라지고 Label에 텍스트가 표시됩니다.



8) 관련 API

Evas_Object *elm_popup_add(Evas_Object *parent) : Popup을 생성하는 API.

void elm_popup_align_set(Evas_Object *obj, double horizontal, double vertical) : Popup의 위치를 지정하는 API. 2번째 파라미터는 수평 위치를 지정하며, ELM_NOTIFY_ALIGN_FILL은 화면 수평 영역 전체를 사용하는 옵션입니다. 3번째 파라미터는 수직 위치를 지정하며 수치 범위는 0.0~1.0 사이입니다.

void evas_object_size_hint_weight_set(Evas_Object *obj, double x, double y) : 크기를 대략적으로 지정하는 API. / 파라미터 : 윈도우 객체, 넓이 힌트, 높이 힌트. EVAS_HINT_EXPAND는 가능한 확장하는 옵션입니다.

void elm_popup_timeout_set(Evas_Object *obj, double timeout) : Popup에 타이머 이벤트를 지정하는 API. 2번째 파라미터는 시간 간격을 지정합니다. 3.0을 지정하면 3초 후에 타이머 이벤트가 발생합니다.

void evas_object_smart_callback_add(Evas_Object *obj, const char *event, Evas_Smart_Cb func, const void *data) : 이벤트를 수신하는 콜백 함수를 지정하는 API. 2번째 파라미터에 'timeout'을 지정하면 타이머 이벤트가 발생할 때 콜백 함수가 호출됩니다. 3번째 파라미터에는 콜백 함수 이름을 지정합니다.

void evas_object_del(Evas_Object *obj) : 객체를 삭제하는 API.

14. Slider 위젯 사용방법

사용자가 수치값을 입력할 때 Slider를 사용하면 편리합니다. 오디오 재생기 혹은 비디오 재생기의 재생 위치 탐색에도 유용하게 사용됩니다.

1) Slider 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 SliderEx으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 sliderex.c 파일을 열고 create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
```

```

EVAS_HINT_EXPAND);

/* fill the expanded area (above) as opposed to center in it */
evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
/* actually put the child in the frame and show it */
evas_object_show(child);
elm_object_content_set(frame, child);

}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);

}

```

그런 다음 `create_base_gui()` 함수로 이동해서 Box 컨테이너와 Slider 위젯을 생성하겠습니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
    /* child object - indent to how relationship */
    /* A box to put things in vertically - default mode for box */
    Evas_Object *box = elm_box_add(ad->win);
    evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
}
```

```

elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "Please test the slider below");
    my_box_pack(box, ad->label, 1.0, 0.1, 0.5, 1.0);

    /* Slider-1 */
    Evas_Object *slider = elm_slider_add(ad->conform);
    elm_slider_min_max_set(slider, 0, 9);
    elm_slider_value_set(slider, 5);
    my_box_pack(box, slider, 1.0, 0.1, -1.0, 0.0);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

elm_slider_add() 는 Slider 위젯을 생성하는 API입니다.

elm_slider_min_max_set() 는 범위를 지정하는 API입니다. 2번째 파라미터는 최소값, 3번째 파라미터는 최대값을 지정하면 됩니다. 변수 타입은 double 입니다.

elm_slider_value_set() 는 현재 값을 지정하는 API입니다.

소스 프로젝트를 빌드하고 실행시켜 봅시다. Slider 위젯이 보입니다.
트래킹바를 좌우로 이동시켜 봅시다.



2) Slider 위젯에 인디케이터 표시

트래킹바를 드래그 할 때 현재 값을 표시하는 기능을 인디케이터라고 합니다. Slider를 생성하는 코드에 새로운 코드를 추가합시다.

```
/* Slider-1 */
Evas_Object *slider = elm_slider_add(ad->conform);
elm_slider_min_max_set(slider, 0, 9);
elm_slider_value_set(slider, 5);
elm_slider_indicator_show_set(slider, EINA_TRUE);
elm_slider_indicator_format_set(slider, "%1.0f");
my_box_pack(box, slider, 1.0, 0.1, -1.0, 0.0);
```

`elm_slider_indicator_show_set()` 는 Slider에 인디케이터 표시여부를 지정하는 API입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 인디케이터가 표시됩니다. EINA_FALSE를 전달하면 반대가 됩니다.

`elm_slider_indicator_format_set()` 는 인디케이터에 표시되는 텍스트의 포맷을 지정하는 API 입니다.

예제를 다시 실행하고 트래킹바를 이동시켜 봅시다. 트래킹바 위에 숫자가 표시됩니다.



3) Slider 트래킹 이벤트

사용자가 트래킹바를 드래그 할 때의 이벤트를 실시간으로 구해 봅시다.
비디오 플레이어를 구현할 때는 이 기능이 필요합니다.

Slider를 생성하는 코드에 새로운 코드를 추가합니다.

```
/* Slider-1 */
Evas_Object *slider = elm_slider_add(ad->conform);
elm_slider_min_max_set(slider, 0, 9);
elm_slider_value_set(slider, 5);
elm_slider_indicator_show_set(slider, EINA_TRUE);
elm_slider_indicator_format_set(slider, "%1.0f");
evas_object_smart_callback_add(slider, "changed", slider_changed_cb, ad);
my_box_pack(box, slider, 1.0, 0.1, -1.0, 0.0);
```

`evas_object_smart_callback_add()` 함수의 2번째 파라미터에 'changed'를 전달하면 Slider의 값이 변경될 때 이벤트를 구할 수 있습니다.

이제 이벤트 함수를 정의할 차례입니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다. Slider의 현재 값을 구해서 Label 위젯에 표시하는 코드입니다.

```
static void
slider_changed_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char buf[64];

    double value = elm_slider_value_get(obj);
    sprintf(buf, "Slider : %d", (int)value);
```

```
    elm_object_text_set(ad->label, buf);
}
```

elm_slider_value_get() 는 elm_slider_value_set() 과 반대 기능의 API입니다. Slider의 현재 값을 반환해 줍니다. 변수 타입은 double입니다.

예제를 다시 실행하고 트래킹바를 이동해 봅시다. Label 위젯에 현재 값이 표시됩니다.



4) 센터 포인트 표시

Slider의 가운데 위치에 센터 포인트를 표시할 수 있습니다.

create_base_gui() 함수에 새로운 Label과 Slider를 생성하는 코드를 추가하겠습니다.

```
/* Slider-1 */
Evas_Object *slider = elm_slider_add(ad->conform);
elm_slider_min_max_set(slider, 0, 9);
elm_slider_value_set(slider, 5);
elm_slider_indicator_show_set(slider, EINA_TRUE);
elm_slider_indicator_format_set(slider, "%1.0f");
evas_object_smart_callback_add(slider, "changed", slider_changed_cb, ad);
my_box_pack(box, slider, 1.0, 0.1, -1.0, 0.0);

/* Label-2 */
ad->label2 = elm_label_add(ad->conform);
elm_object_text_set(ad->label2, "Please test the slider below");
```

```

my_box_pack(box, ad->label2, 1.0, 0.1, 0.5, 1.0);

/* Slider-2 */
slider = elm_slider_add(ad->conform);
elm_slider_min_max_set(slider, 0, 99);
elm_slider_value_set(slider, 30);
elm_object_style_set(slider, "center_point");
evas_object_smart_callback_add(slider, "changed", slider2_changed_cb, ad);
my_box_pack(box, slider, 1.0, 0.1, -1.0, 0.0);

}

```

elm_object_style_set() 함수 1번째 파라미터에 Slider 위젯을 전달하고, 2번째 파라미터에 'center_point'를 전달하였습니다. 이렇게 하면 Slider 위젯 중앙에 눈금이 표시됩니다.

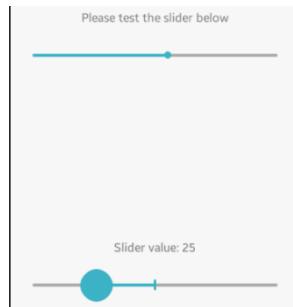
2번째 Slider의 콜백 함수를 정의하겠습니다. create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```

static void
slider2_changed_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char buf[64];

    double value = elm_slider_value_get(obj);
    sprintf(buf, "Slider value: %d", (int)value);
    elm_object_text_set(ad->label2, buf);
}
```

예제를 다시 실행하면 2번째 Slider 위젯이 생성되었고 센터 포인트가 표시되었습니다. 트래킹바를 드래그하면 2번째 Label에 Value가 표시됩니다.



5) 관련 API

Evas_Object *elm_slider_add(Evas_Object *parent) : Slider 위젯을 생성하는 API.

void elm_slider_min_max_set(Evas_Object *obj, double min, double max) : Slider 위젯의 범위를 지정하는 API. 파라미터 : Slider 위젯 객체, 최소값, 최대값.

void elm_slider_value_set(Evas_Object *obj, double val) : Slider 위젯의 현재 값을 지정하는 API.

void elm_slider_indicator_show_set(Evas_Object *obj, Eina_Bool show) : Slider에 인디케이터 표시여부를 지정하는 API. 2번째 파라미터에 EINA_TRUE를 전달하면 인디케이터가 표시됩니다. EINA_FALSE를 전달하면 반대가 됩니다.

void elm_slider_indicator_format_set(Evas_Object *obj, const char *indicator) : 인디케이터에 표시되는 텍스트의 포맷을 지정하는 API. /
파라미터 - Slider 객체, 텍스트 포맷.

double elm_slider_value_get(const Evas_Object *obj) : Slider의 현재 값을
반환하는 API. 변수 타입은 double.

Eina_Bool elm_object_style_set(Evas_Object *obj, const char *style) :
오브젝트의 스타일을 지정하는 API. 1번째 파라미터에 Slider 위젯을
전달하고, 2번째 파라미터에 'center_point'를 전달하면 Slider 위젯
중앙에 눈금이 표시됩니다.

15. List 위젯에 텍스트 Item 추가

여러개의 텍스트 목록을 화면에 표시하려면 List 위젯을 사용하면 됩니다. 위아래로 스크롤 할 수 있고, 사용자의 선택 이벤트를 구할 수도 있습니다. 예제를 통해서 List 위젯의 사용방법을 알아보겠습니다.

1) 텍스트 List 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 ListEx으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",  
     * "pad_large" and "pad_huge" available as styles in addition to the  
     * "default" frame style */  
    elm_object_style_set(frame, "pad_medium");  
    /* set the input weight/aling on the frame insted of the child */  
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);  
    evas_object_size_hint_align_set(frame, h_align, v_align);  
    /* tell the child that is packed into the frame to be able to expand */
```

```

    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

    /* fill the expanded area (above) as opposed to center in it */
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    /* actually put the child in the frame and show it */
    evas_object_show(child);
    elm_object_content_set(frame, child);

}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);

}

```

그런 다음 `create_base_gui()` 함수로 이동해서 Box 컨테이너와 List 위젯을 생성하겠습니다. 그런 다음 10개의 텍스트 항목을 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
    /* child object - indent to how relationship */
    /* A box to put things in vertically - default mode for box */
    Evas_Object *box = elm_box_add(ad->win);
    evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,

```

```

EVAS_HINT_EXPAND);

elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

/* List */
const char *items[] = { "Seoul", "Tokyo", "Newyork", "Londeon",
"Baijing", "Kongga", "Moscuba", "Singgapol", "Pusan", "Hongkong" };
Evas_Object *list = elm_list_add(ad->conform);

for(int i=0; i < 10; i++)
    elm_list_item_append(list, items[i], NULL, NULL, NULL, (void*)i);
elm_list_go(list);
my_box_pack(box, list, 1.0, 1.0, -1.0, -1.0);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

elm_list_add() 는 새로운 List 위젯을 추가하는 API입니다.

elm_list_item_append() 는 List 위젯에 항목을 추가하는 API입니다. 1번째 파라미터는 List 위젯의 객체이고, 2번째는 텍스트 문자열입니다. 3번째는 왼쪽 아이콘, 4번째 파라미터는 오른쪽 아이콘입니다. 5번째는 항목 선택 이벤트 함수를 콜백 스타일로 지정합니다. 6번째는 사용자 데이터를

전달합니다. 일반적으로 appdata를 전달하지만 몇 번째 항목인지를 알기 위해서 인덱스 번호를 전달하였습니다.

elm_list_go() 는 화면을 갱신해서 변경된 항목을 화면에 표시하는 API입니다. 새로운 항목을 추가하거나 항목 삭제 혹은 수정했을 경우 이 함수를 호출해 주어야 화면에 결과가 보여지는 것입니다.

예제를 실행하면 화면에 List 위젯이 생성되고 10개의 텍스트가 표시됩니다. 위아래로 드래그해서 목록을 스크롤 할 수 있습니다. 항목을 선택하면 선택 표시가 나타납니다.



2) 선택 표시 자동 해제

항목을 선택했을 때 선택 표시가 사라지지 않고 남아있습니다. 이 기능이 필요한 경우도 있지만 그렇지 않은 경우도 있습니다. 이 기능을 해제하는 방법을 알아보겠습니다. create_base_gui() 함수에 새로운 코드 1줄을 추가합니다.

```
    elm_list_go(list);
    evas_object_smart_callback_add(list, "selected", list_selected_cb, NULL);
    my_box_pack(box, list, 1.0, 1.0, -1.0, -1.0);
}

```

사용자가 List 위젯 항목을 선택했을 때 호출되는 콜백 함수를 지정하는 코드입니다. 콜백 함수를 정의해 주어야 한다는 의미입니다.

create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void
list_selected_cb(void *data, Evas_Object *obj, void *event_info)
{
    Elm_Object_Item *it = event_info;
    elm_list_item_selected_set(it, EINA_FALSE);
}

```

elm_list_item_selected_set() 는 List 위젯 항목 선택 표시를 지정/해제 하는 API입니다. 1번째 파라미터는 항목의 객체를 전달하면 되는데, 콜백 함수 3번째 파라미터에 넘어오는 event_info를 전달하면 됩니다. 2번째 파라미터에 EINA_TRUE를 전달하면 선택 표시가 나타나고, EINA_FALSE를 전달하면 선택 표시가 사라집니다.

다시 실행해서 List 위젯 항목을 선택해 봅시다. 이번에는 선택표시가 나타났다가 사라집니다.

3) List 위젯 항목 선택 이벤트 구하기

사용자가 List 위젯 항목을 선택하면 해당 항목의 인덱스 번호와 텍스트를 구해서 Label 위젯에 표시하는 기능을 구현해 보겠습니다.

List 위젯에 항목을 추가하는 코드를 아래와 같이 수정합니다.

```
for(int i=0; i < 10; i++)
    elm_list_item_append(list, items[i], NULL, NULL, list_item_clicked, (void*)i);
//elm_list_item_append(list, items[i], NULL, NULL, NULL, (void*)i);
```

항목 선택 이벤트 콜백 함수를 `list_item_clicked`로 지정하였습니다. 이제 이 함수를 정의할 차례입니다.

`create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void
list_item_clicked(void *data, Evas_Object *obj, void *event_info)
{
    int index = (int)data;
    Elm_Object_Item *it = event_info;
    const char *item_text = elm_object_item_text_get(it);

    char buf[PATH_MAX];
    sprintf(buf, "%d - %s", index, item_text);
    dlog_print(DLOG_INFO, "tag", "%s", buf);
}
```

사용자가 List 위젯의 항목을 선택하면 위 함수가 호출됩니다. 1번째 파라미터에는 항목 인덱스 번호가 전달됩니다. 2번째 파라미터에는 List 위젯의 객체가 전달되고, 3번째 파라미터에는 선택된 항목의 객체가 전달됩니다.

elm_object_item_text_get() 는 항목의 텍스트를 반환하는 API입니다.

그 이후는 인덱스 번호와 항목 텍스트를 문자열 변수에 저장해서 로그창에 표시하는 코드입니다.

예제를 다시 빌드하고 실행시켜 봅시다. 그리고 항목을 선택하면 Log 패널에 항목 정보가 표시됩니다. 이클립스 아래쪽에 Log 패널이 보이지 않을 때는 메뉴 [Window > Show View > Other...]를 선택하고 팝업창에서 [Tizen > Log]를 선택하면 됩니다.

로그 메시지를 보려면 이클립스 Log 패널에서 아래쪽 콤보박스에 Tag를 선택하고, 오른쪽 에디트박스에 tag라고 입력합니다.

Time	Level	Pid	Tid	Tag	Message
06-08 13:29:38.639	Info	3742	3742	tag	1 - Tokyo
06-08 13:29:38.639	Error	3742	3742	tag	list_selected_cb
06-08 13:29:50.049	Info	3742	3742	tag	3 - Londeon
06-08 13:29:50.049	Error	3742	3742	tag	list_selected_cb
06-08 13:29:55.949	Info	3742	3742	tag	5 - Kongga
06-08 13:29:55.959	Error	3742	3742	tag	list_selected_cb

Tag tag

4) Label에 항목 정보 표시하기

항목 선택 이벤트 함수에 appdata 대신에 인덱스 번호를 전달하였습니다. Label 위젯에 정보를 표시하려면 appdata를 전역 변수로 선언하면 됩니다.

소스파일 위 부분에 새로운 코드 한줄을 추가합니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
} appdata_s;
```

```
appdata_s* m_ad = 0;
```

appdata를 전역 변수로 선언하는 코드입니다.

create_base_gui() 함수 시작 부분에서 아래와 같이 초기화 해줍니다.

```
static void  
create_base_gui(appdata_s *ad)  
{  
    m_ad = ad;
```

이제 어디서나 Label 위젯을 사용할 수 있습니다.

list_item_clicked() 함수로 되돌아가서 함수 끝부분에 아래와 같이 코드 한줄을 추가합니다.

```
dlog_print(DLOG_INFO, "tag", "%s", buf);
elm_object_text_set(m_ad->label, buf);
}
```

Label 위젯에 선택된 List 위젯 항목의 정보를 표시하는 코드입니다.

예제를 다시 실행한 다음 List 위젯의 항목을 선택해 봅시다. 선택된 항목의 정보가 Label 위젯에 표시됩니다.



5) 관련 API

Evas_Object *elm_list_add(Evas_Object *parent) : List 위젯 추가 API. /
파라미터 - 부모 객체.

Elm_Object_Item *elm_list_item_append(Evas_Object *obj, const char
*label, Evas_Object *icon, Evas_Object *end, Evas_Smart_Cb func, const
void *data) : List 위젯에 항목 추가 API. / 파라미터 - List 위젯, 항목
텍스트, 왼쪽 아이콘, 오른쪽 아이콘, 항목 선택 이벤트 함수명, 사용자
데이터.

`void elm_list_go(Evas_Object *obj)` : 목록을 시작하는 API. List 위젯을 화면에 표시하기 전에 이 함수가 먼저 호출되어야 합니다. 새로운 항목을 추가하거나 삭제, 수정할 때에 이 함수를 호출해 주어야 결과가 화면에 표시됩니다. / 파라미터 - List 위젯.

`void elm_list_item_selected_set(Elm_Object_Item *it, Eina_Bool selected)` : 항목 선택표시를 지정&해제하는 API. / 파라미터 - List 위젯, 선택표시 지정 여부.

`const char *elm_object_item_text_get(const Elm_Object_Item *it)` : 항목 텍스트를 반환하는 API. / 파라미터 - 항목 객체.

16. GenList 위젯에 아이콘 표시

List 위젯에 아이콘을 표시하거나 2줄의 텍스트를 표시하려면 GenList 위젯을 사용하면 됩니다. 목록을 그룹으로 구분할 수도 있습니다.

GenList 위젯은 항목을 추가할 때와 삭제할 때 그리고 아이콘 같은 콘텐츠를 추가할 때 콜백 함수에서 처리해야 하며, 항목 데이터를 저장하기 위해서 구조체를 사용해야 하기 때문에 List 위젯 보다는 사용 방법이 번거롭습니다. 예제를 통해서 구체적인 사용방법을 알아보겠습니다.

1) GenList 위젯 생성 & 텍스트 표시

새로운 소스 프로젝트를 생성하고 Project name을 GenListEx으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 위 부분에 새로운 구조체를 정의합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
} appdata_s;
```

```
typedef struct item_data
{
    int index;
    Elm_Object_Item *item;
} item_data_s;
```

item_data 는 GenList 위젯 항목 데이터를 저장하는 구조체입니다.

create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",  
     * "pad_large" and "pad_huge" available as styles in addition to the  
     * "default" frame style */  
    elm_object_style_set(frame, "pad_medium");  
    /* set the input weight/aling on the frame insted of the child */  
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);  
    evas_object_size_hint_align_set(frame, h_align, v_align);  
    {  
        /* tell the child that is packed into the frame to be able to expand */  
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,  
                                         EVAS_HINT_EXPAND);  
        /* fill the expanded area (above) as opposaed to center in it */  
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);  
        /* actually put the child in the frame and show it */  
        evas_object_show(child);  
        elm_object_content_set(frame, child);  
    }  
    /* put the frame into the box instead of the child directly */  
    elm_box_pack_end(box, frame);  
    /* show the frame */  
    evas_object_show(frame);  
}
```

그런 다음 `create_base_gui()` 함수로 이동해서 Box와 List 위젯을 생성하겠습니다. 그런 다음 10개의 텍스트 항목을 추가합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

/* Genlist */
Evas_Object *genlist = elm_genlist_add(ad->conform);
my_box_pack(box, genlist, 1.0, 1.0, -1.0, -1.0);

/* Create item class */
Elm_Genlist_Item_Class *itc = elm_genlist_item_class_new();
```

```

itc->item_style = "end_icon";
itc->func.text_get = gl_text_get_cb;
itc->func.del = gl_del_cb;

/* Item add */
for(int i=0; i < 10 ; i++)
{
    item_data_s *id = calloc(sizeof(item_data_s), 1);
    id->index = i;
    id->item = elm_genlist_item_append(genlist, itc, id, NULL,
ELM_GENLIST_ITEM_NONE, NULL, id);
}

elm_genlist_item_class_free(itc);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Box를 생성한 다음, GenList를 생성해서 Box에 추가하였습니다.

`elm_genlist_add()` 는 GenList 위젯을 생성하는 API 입니다.

`elm_genlist_item_class_new()` 는 GenList 항목 클래스를 생성하는 API입니다.

`Elm_Genlist_Item_Class` 는 GenList 항목 클래스입니다. 이 클래스를 사용해서 GenList 스타일과 콜백 함수를 지정합니다. 속성 종류는 다음과 같습니다.

- `item_style` 속성에 "end_icon"를 전달하면 오른쪽 끝에 아이콘이 표시됩니다.

- func.text_get 는 항목 텍스트를 지정하는 콜백 함수를 대입합니다.
- func.del 는 항목을 삭제하는 콜백 함수를 대입합니다.

elm_genlist_item_append() 는 GenList에 항목을 추가하는 API입니다. 이 함수를 호출하면 func.text_get 속성에 대입한 콜백 함수가 자동 호출됩니다. 파라미터는 순서대로 GenList 객체, 항목 클래스, 항목 데이터, 부모 항목, 항목 종류, 항목 선택 이벤트 콜백 함수명, 사용자 데이터 입니다.

elm_genlist_item_class_free() 는 GenList 항목 클래스를 삭제하는 API입니다.

이제 GenList 위젯 항목 텍스트를 지정하는 콜백 함수를 정의합시다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static char*
gl_text_get_cb(void *data, Evas_Object *obj, const char *part)
{
    const char *items[] = { "Seoul", "Tokyo", "Newyork", "Londeon", "Baijing", "Kongga",
    "Moscuba", "Singgapol", "Pusan", "Hongkong" };
    item_data_s *id = data;

    if (!strcmp(part, "elm.text")) {
        return strdup(items[id->index]);
    }

    return NULL;
}
```

위 코드는 GenList 위젯 항목에 데이터를 입력하는 콜백 함수입니다. 1번째 파라미터에는 사용자 데이터가 전달됩니다. 여기서는 항목 데이터

구조체가 전달됩니다. index 속성에는 항목 인덱스 번호가 저장되어 있습니다. 2번째 파라미터는 항목의 객체입니다. 3번째 파라미터는 엘리먼트의 종류가 전달됩니다.

하나의 GenList 항목은 여러개의 엘리먼트를 가질수 있습니다. 이 값이 "elm.text" 이면 텍스트 엘리먼트인 것입니다.

strcmp (char *, char *) 는 2개의 문자열을 비교하는 API입니다. 같은 문자열이면 0을 반환합니다.

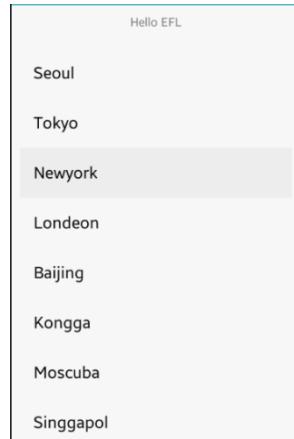
위 함수는 항목 인덱스 번호에 해당하는 텍스트를 반환하면 되는데 strdup() 함수를 사용하는 이유는 새로운 문자열을 생성하기 위해서입니다. 그렇게하지 않으면 함수가 종료됨과 동시에 문자열 데이터도 함께 사라집니다.

이제 항목이 삭제될 때 호출되는 콜백 함수를 정의합시다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void  
gl_del_cb(void *data, Evas_Object *obj)  
{  
    item_data_s *id = data;  
    free(id);  
}
```

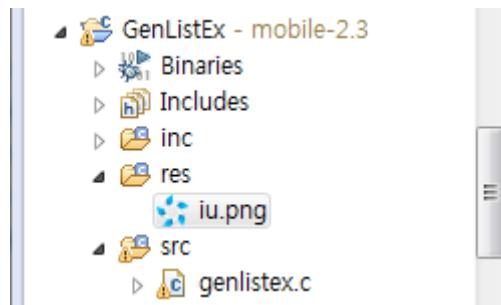
1번째 파라미터에는 사용자 데이터가 전달됩니다. 항목 데이터 구조체이니 이것을 삭제해 주면 됩니다.

예제를 빌드하고 실행시켜 봅시다. GenList가 생성되었고, 10개의 텍스트 항목이 추가되었습니다.



2) GenList에 아이콘 표시

항목 오른쪽에 아이콘 이미지를 표시해 보겠습니다. 그러기 위해서 이미지 파일이 필요합니다. 부록 /Image/iu.png 파일을 소스 프로젝트 /res 폴더로 복사합니다.



그런 다음 GenList 항목에 아이콘 이미지를 추가하는 콜백함수를 생성합니다. `create_base_gui()` 함수 위에 아래와 같이 3개의 함수를 추가합니다.

```

static void
app_get_resource(const char *res_file_in, char *res_path_out, int res_path_max)
{
    char *res_path = app_get_resource_path();
    if (res_path) {
        snprintf(res_path_out, res_path_max, "%s%s", res_path, res_file_in);
        free(res_path);
    }
}

static Evas_Object*
create_image(Evas_Object *parent)
{
    char img_path[PATH_MAX] = { 0, };
    app_get_resource("iu.png", img_path, PATH_MAX);
    Evas_Object *img = elm_image_add(parent);
    elm_image_file_set(img, img_path, NULL);
    return img;
}

static Evas_Object*
gl_content_get_cb(void *data, Evas_Object *obj, const char *part)
{
    Evas_Object *content = create_image(obj);
    evas_object_size_hint_min_set(content, 50, 50);
    evas_object_size_hint_max_set(content, 50, 50);
    return content;
}

```

app_get_resource() 는 파일명에 res 폴더 경로를 추가해서 반환하는 함수입니다.

`app_get_resource_path()` 는 res 폴더의 절대 경로를 반환해 주는 API입니다.

`create_image()` 는 이미지 파일이 적용된 Image 객체를 생성하는 함수입니다.

`elm_image_add()` Image 객체를 생성하는 API입니다.

`elm_image_file_set()` 는 Image 객체에 이미지 파일 경로를 지정해서 이미지를 로딩하는 API 입니다.

`gl_content_get_cb()` 는 GenList 항목에 아이콘 이미지를 지정하는 콜백 함수입니다.

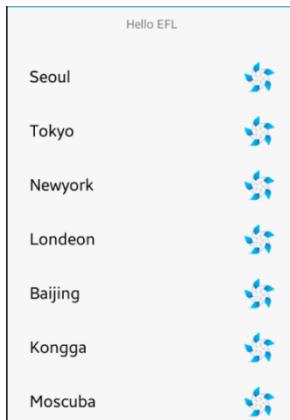
`evas_object_size_hint_min_set()` 는 객체의 최소 크기에 대한 힌트를 지정하는 API입니다. 여기서는 절대값 50을 지정하였습니다.

`evas_object_size_hint_max_set()` 는 객체의 최대 크기에 대한 힌트를 지정하는 API입니다. 여기서는 절대값 50을 지정하였습니다.

방금 만든 아이콘 지정 콜백 함수를 GenList 항목에 대입합시다.
`create_base_gui()` 함수로 이동해서 새로운 코드 한줄을 추가합니다.

```
Elm_Genlist_Item_Class *itc = elm_genlist_item_class_new();
itc->item_style = "end_icon";
itc->func.text_get = gl_text_get_cb;
itc->func.del = gl_del_cb;
itc->func.content_get = gl_content_get_cb;
```

다시 실행하면 GenList 항목 오른쪽에 아이콘이 표시됩니다.



3) 선택 항목 인덱스 번호 구하기

사용자가 항목을 선택하면 해당 항목의 인덱스 번호를 Label 위젯에 표시하는 기능을 구현해 보겠습니다. 소스파일 위부분에 1개의 전역변수와 1개의 함수를 추가합니다.

```
typedef struct item_data
{
    int index;
    Elm_Object_Item *item;
} item_data_s;

appdata_s* m_ad = 0;

static void
list_item_clicked(void *data, Evas_Object *obj, void *event_info)
{
    item_data_s *id = data;
    char buf[PATH_MAX];
    sprintf(buf, "Item-%d", id->index);
```

```
    elm_object_text_set(m_ad->label, buf);  
}  
_____
```

m_ad 는 appdata를 저장하는 전역변수 입니다.

list_item_clicked() 항목 선택 이벤트 콜백 함수이고, 해당 항목 텍스트를 Label 위젯에 표시하는 기능을 담당합니다. 1번째 파라미터에 항목 데이터가 전달됩니다. index 속성에는 인덱스 번호가 저장되어 있습니다. 이것을 문자열로 변경해서 Label 위젯에 표시합니다.

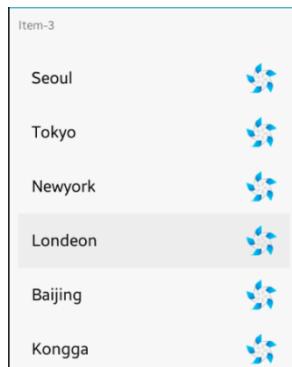
전역변수에 appdata를 저장합시다. create_base_gui() 함수 시작 부분에 코드 한줄을 추가합니다.

```
create_base_gui(appdata_s *ad)  
{  
    m_ad = ad;  
_____
```

이제 콜백 함수만 지정하면 됩니다. create_base_gui() 함수의 코드를 수정합니다.

```
for(int i=0; i < 10 ; i++)  
{  
    item_data_s *id = calloc(sizeof(item_data_s), 1);  
    id->index = i;  
    id->item = elm_genlist_item_append(genlist, itc, id, NULL,  
ELM_GENLIST_ITEM_NONE, list_item_clicked, id);  
    //id->item = elm_genlist_item_append(genlist, itc, id, NULL,  
ELM_GENLIST_ITEM_NONE, NULL, id);  
}_____
```

항목 선택 이벤트 콜백 함수를 `list_item_clicked` 으로 지정하였고,
GenList 클릭 이벤트 콜백 함수를 `gl_selected_cb` 으로 지정하였습니다.
예제를 다시 실행시켜 봅시다. 항목을 선택하면 해당 항목의 인덱스
번호가 Label 위젯에 표시됩니다.



4) 관련 API

`Evas_Object *elm_genlist_add(Evas_Object *parent)` : GenList 위젯 추가
API. / 파라미터 - 부모 객체.

`Elm_Genlist_Item_Class *elm_genlist_item_class_new(void)` : 항목 클래스
생성 API.

`Elm_Genlist_Item_Class` : GenList 항목 클래스.

- `item_style` : 항목 스타일
- `func.text_get` : 항목 텍스트 지정 콜백 함수
- `func.content_get` : 항목 아이콘 지정 콜백 함수
- `func.del` : 항목 삭제 이벤트 콜백 함수

`Elm_Object_Item *elm_genlist_item_append(Evas_Object *obj,`

`Elm_Genlist_Item_Class *itc, void *data, Elm_Object_Item *parent, Elm_Genlist_Item_Type type, Evas_Smart_Cb func, void *func_data)` : GenList에 항목을 추가하는 API. 이 함수를 호출하면 `func.text_get` 속성에 대입한 콜백 함수가 자동 호출됩니다. / 파라미터 - GenList 객체, 항목 클래스, 항목 데이터, 부모 항목, 5번째 파라미터는 항목 종류, 항목 선택 이벤트 콜백 함수명, 사용자 데이터.

`void elm_genlist_item_class_free(Elm_Genlist_Item_Class *itc)` : GenList 항목 클래스 삭제 API. / GenList 항목 클래스 객체

`char *app_get_resource_path(void)` : res 폴더의 절대경로 반환 API.

`sprintf(char*, char*, ...)` : 포맷 형식으로 문자열 생성 API.

`snprintf(char*, int, char*, ...)` : 길이를 지정하여 포맷 형식으로 문자열 생성 API.

`Evas_Object *elm_image_add(Evas_Object *parent)` : Image 객체를 생성 API. / 부모 객체

`Eina_Bool elm_image_file_set(Evas_Object *obj, char *file, char *group)` : Image 객체에 이미지 파일을 지정하는 API. / 파라미터 - Image 객체, 이미지 파일 경로, 이미지 그룹명(Edje 파일인 경우)

`void evas_object_size_hint_min_set(Evas_Object *obj, Evas_Coord w, Evas_Coord h)` : 객체의 최소 크기 힌트를 지정하는 API. / 파라미터 - 객체, 넓이 힌트, 높이 힌트.

`void evas_object_size_hint_max_set(Evas_Object *obj, Evas_Coord w, Evas_Coord h)` : 객체의 최대 크기 힌트를 지정하는 API. / 파라미터 - 객체, 넓이 힌트, 높이 힌트.

strcmp(char*, char*) : 2개의 문자열 대소를 비교하는 API. 같으면 0을 반환합니다.

strdup(char*) : 동일한 문자열을 하나 더 생성해서 반환하는 API.

void elm_genlist_item_selected_set(Elm_Object_Item *it, Eina_Bool selected) : 항목 선택표시를 지정&해제하는 API. / 파라미터 - List 위젯, 선택표시 지정 여부.

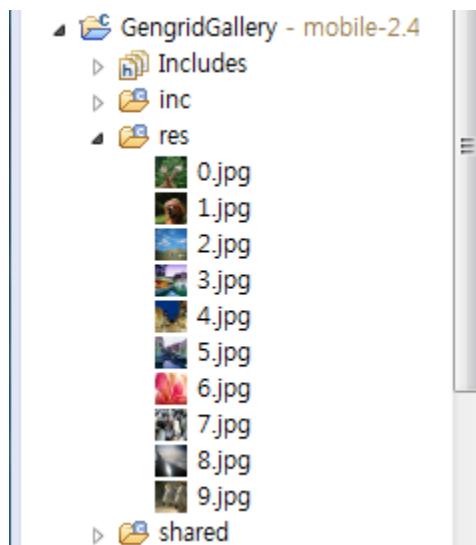
17. Gallery 복합 위젯 만들기

2개 이상의 위젯을 연동하여 복합 위젯을 만드는 방법을 알아보겠습니다. 이번 예제에서는 GenGrid 위젯과 Bg 위젯을 사용해서 갤러리 위젯을 제작해 보겠습니다.

1) GenGrid 위젯 생성 & 이미지 추가

새로운 소스 프로젝트를 생성하고 Project name을 GengridGallery으로 지정합니다.

이번 예제는 이미지를 보여주는 갤러리 위젯을 제작할 것입니다. 따라서 이미지 파일이 필요합니다. 부록 /Image 폴더에서 0.jpg ~ 9.jpg 총 10개의 파일을 소스 프로젝트 /res 폴더로 복사합니다.



/src 폴더에서 소스파일(~.c)을 열고 appdata 구조체에 새로운 변수를

추가합니다. 그리고 새로운 구조체를 정의합니다.

```
-----  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    Evas_Object *gengrid;  
    Evas_Object *bg;  
} appdata_s;
```

```
-----  
typedef struct itemdata {  
    int index;  
    const char *path;  
} itemdata_s;
```

AppData 구조체에 GenGrid, Bg 변수를 추가하였습니다.

itemdata는 GenGrid 항목 정보를 저장하는 구조체입니다. 항목 정보는 인덱스 번호와 이미지 파일의 경로입니다.

GenGrid 위젯을 생성하고 이미지를 추가해 보겠습니다. 사용 방법은 GenList 위젯과 유사합니다. create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
-----  
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */
```

```

Evas_Object *frame = elm_frame_add(box);
/* use the medium padding style. there is "pad_small", "pad_medium",
 * "pad_large" and "pad_huge" available as styles in addition to the
 * "default" frame style */
elm_object_style_set(frame, "pad_medium");
/* set the input weight/aling on the frame insted of the child */
evas_object_size_hint_weight_set(frame, h_weight, v_weight);
evas_object_size_hint_align_set(frame, h_align, v_align);
{
    /* tell the child that is packed into the frame to be able to expand */
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    /* fill the expanded area (above) as opposaed to center in it */
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    /* actually put the child in the frame and show it */
    evas_object_show(child);
    elm_object_content_set(frame, child);
}
/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}

```

create_base_gui() 함수로 이동해서 Box, GenGrid, Bg 위젯을 생성하는 코드를 추가합니다. Label은 이번 예제에서 사용하지 않으므로 삭제합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);

```

```

evas_object_size_hint_weight_set(ad->conform,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* Gengrid */
ad->gengrid = create_gengrid(ad->conform);
my_box_pack(box, ad->gengrid, 1.0, 1.0, -1.0, -1.0);

/* Bg-1 Color */
ad->bg = elm_bg_add(ad->conform);
elm_bg_color_set(ad->bg, 66, 162, 206);
my_box_pack(box, ad->bg, 1.0, 1.0, -1.0, -1.0);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

create_gengrid() 는 GenGrid 위젯을 생성해서 반환하는 함수이고,
잠시후 만들어 보겠습니다.

elm_bg_color_set() 는 Bg 위젯의 배경 컬러를 지정하는 함수입니다.

이번 예제에서는 필요없는 기능이지만 Bg의 위치를 확인하기 위해서 사용하였습니다. 예제가 완성되면 주석처리해도 상관없습니다.

GenGrid 위젯을 생성하는 함수를 정의하겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static Evas_Object*
create_gengrid(appdata_s *ad)
{
    Elm_Gengrid_Item_Class *gic;
    Evas_Object *gengrid;
    char buf[PATH_MAX];

    gengrid = elm_gengrid_add(ad->conform);
    elm_gengrid_item_size_set(gengrid, ELM_SCALE_SIZE(60), ELM_SCALE_SIZE(60));
    elm_gengrid_horizontal_set(gengrid, EINA_TRUE);

    gic = elm_gengrid_item_class_new();
    gic->func.content_get = gengrid_content_get_cb;

    for(int i = 0; i < 10; i++) {
        itemdata_s *id = calloc(sizeof(itemdata_s), 1);
        snprintf(buf, sizeof(buf), "%s%d.jpg", app_get_resource_path(), i);
        id->index = i;
        id->path = strdup(buf);
        elm_gengrid_item_append(gengrid, gic, id, NULL, id);
    }

    return gengrid;
}
```

`elm_gengrid_add()` 는 GenGrid 위젯을 생성하는 API입니다.

`elm_gengrid_item_size_set()` 는 GenGrid의 아이콘 항목 크기를 지정하는 API입니다.

`elm_gengrid_horizontal_set()` 는 슬라이드 방향을 수평으로 지정&해제하는 API입니다. 기본 설정은 수직 방향입니다.

`elm_gengrid_item_class_new()` 는 GenGrid 항목 정보 클래스를 생성하는 API입니다.

`Elm_Gengrid_Item_Class`는 GenGrid 항목 정보 클래스입니다. 클래스 속성 종류는 다음과 같습니다.

- `func.content_get` : 아이콘 항목을 지정하는 콜백 함수명 대입.
- `item_style` : 항목 스타일 지정. 기본 설정은 "default".
- `func.text_get` : 텍스트 항목을 지정하는 콜백 함수명 대입.
- `func.del` : 항목 삭제 이벤트 콜백 함수명 대입. 이 함수에서 데이터를 삭제하면 됩니다.

`app_get_resource_path()` 는 /res 폴더의 절대 경로를 반환하는 API입니다.

`elm_gengrid_item_append()` 는 GenGrid에 새로운 항목을 추가하는 API입니다. 파라미터는 순서대로 GenGrid 객체, 항목 클래스, 항목 데이터 구조체, 항목 선택 이벤트 콜백 함수명, 사용자 데이터 입니다.

GenGrid 항목에 아이콘 이미지를 추가하는 함수를 정의하겠습니다. `create_gengrid()` 함수 위에 새로운 함수를 추가합니다. 항목이 생성되면 자동으로 이 함수가 호출됩니다.

```
static Evas_Object*
gengrid_content_get_cb(void *data, Evas_Object *obj, const char *part)
```

```

{
    itemdata_s *id = data;

    if (!strcmp(part, "elm.swallow.icon")) {
        Evas_Object *img = elm_image_add(obj);

        elm_image_file_set(img, id->path, NULL);
        elm_image_aspect_fixed_set(img, EINA_FALSE);
        evas_object_show(img);
        return img;
    }
    return NULL;
}

```

위 함수의 1번째 파라미터는 사용자 데이터이고, 2번째 파라미터는 항목 객체입니다. 3번째 파라미터는 엘리먼트의 종류가 전달됩니다. GenGrid 위젯의 항목은 여러개의 엘리먼트로 구성되는데 엘리먼트 종류가 "elm.swallow.icon" 일때 아이콘 이미지를 생성하면 됩니다.

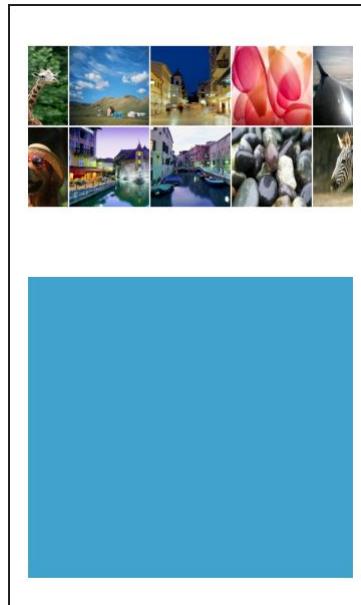
`elm_image_add()` 는 이미지 객체를 생성하는 API입니다.

`elm_image_file_set()` 는 이미지 객체에 파일 경로를 지정해서 이미지를 로딩하는 API입니다.

`elm_image_aspect_fixed_set()` 는 이미지 잘림 여부를 지정하는 API입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 이미지에 잘리는 부분이 없도록 축소됩니다. EINA_FALSE를 전달하면 항목 영역에 이미지를 꽉 채우기 위해서 잘리는 영역이 발생합니다. 기본 설정은 EINA_TRUE 입니다.

예제를 빌드하고 실행시켜 봅시다. 화면 위쪽에 10개의 아이콘 이미지가 표시됩니다. 좌우로 드래그 해봅시다. 이미지 목록이 스크롤 됩니다.

아래쪽에 보이는 사각형은 Bg 위젯 입니다.



2) 선택 아이콘을 Bg에 표시

사용자가 GenGrid에서 아이콘을 선택하면 해당 이미지를 Bg 위젯에 표시하는 기능을 구현해 보겠습니다.

소스파일 위쪽에 전역변수를 하나 추가합니다. appdata를 저장하는 전역변수입니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    Evas_Object *gengrid;  
    Evas_Object *bg;  
} appdata_s;
```

```
appdata_s* m_ad = 0;
```

```
_____
```

그런 다음 `create_base_gui()` 함수 시작 부분에서 전역변수를 초기화 합니다.

```
_____
```

```
static void  
create_base_gui(appdata_s *ad)  
{  
    m_ad = ad;
```

```
_____
```

사용자가 `GenGrid` 항목을 선택했을 때 이벤트 콜백 함수를 생성하겠습니다. `create_gengrid()` 함수 끝부분에 코드를 다음과 같이 수정합니다. 항목 선택 이벤트 콜백 함수명을 `gengrid_it_cb`로 지정하였습니다.

```
_____
```

```
for(int i = 0; i < 10; i++) {  
    itemdata_s *id = calloc(sizeof(itemdata_s), 1);  
    snprintf(buf, sizeof(buf), "%s%d.jpg", app_get_resource_path(), i);  
    id->index = i;  
    id->path = strdup(buf);  
    elm_gengrid_item_append(gengrid, gic, id, gengrid_it_cb, id);  
    //elm_gengrid_item_append(gengrid, gic, id, NULL, id);  
}  
  
return gengrid;  
}
```

```
_____
```

`create_gengrid()` 함수 위에 새로운 함수를 추가합니다. 항목 선택 이벤트

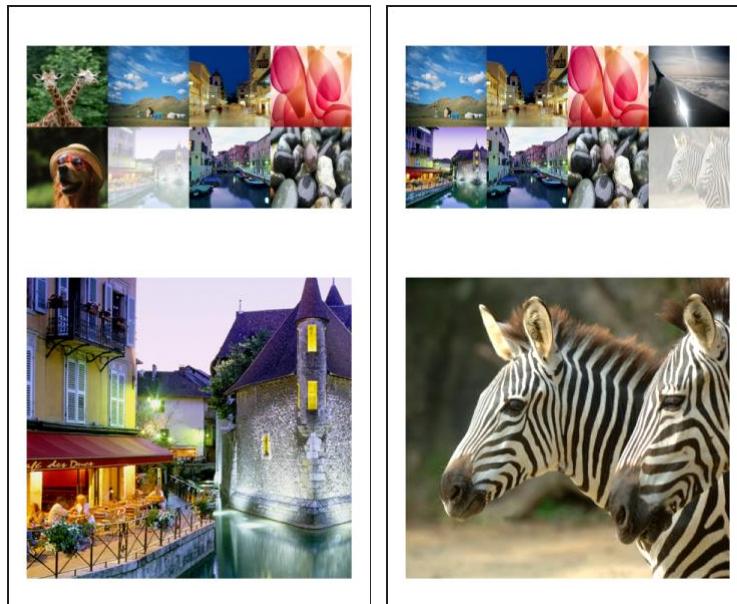
콜백 함수입니다.

```
static void  
gengrid_it_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    itemdata_s *id = data;  
    elm_bg_file_set(m_ad->bg, id->path, NULL);  
}
```

함수의 1번째 파라미터는 항목 데이터 구조체 입니다. path 속성에는 이미지 파일의 경로가 저장되어 있습니다.

elm_bg_file_set() 은 Bg 위젯에 배경 이미지 파일을 지정하는 API입니다.

예제를 다시 실행시켜서 항목 아이콘을 선택해 봅시다. 해당 이미지가 Bg 위젯에 표시됩니다.



3) 관련 API

Elm_Gengrid_Item_Class : GenGrid 항목 클래스. 클래스 속성 종류는 다음과 같습니다.

- func.content_get : 아이콘 항목을 지정하는 콜백 함수명 대입.
- item_style : 항목 스타일 지정. 기본 설정은 "default".
- func.text_get : 텍스트 항목을 지정하는 콜백 함수명 대입.
- func.del : 항목 삭제 이벤트 콜백 함수명 대입. 이 함수에서 데이터를 삭제.

Evas_Object *elm_gengrid_add(Evas_Object *parent) : GenGrid 위젯을 생성하는 API. / 파라미터 - 부모 객체.

void elm_gengrid_item_size_set(Evas_Object *obj, Evas_Coord w, Evas_Coord h) : GenGrid의 아이콘 항목 크기를 지정하는 API. / 파라미터 - GenGrid 객체, 넓이, 높이.

void elm_gengrid_horizontal_set(Evas_Object *obj, Eina_Bool horizontal) : GenGrid 슬라이드 방향을 수평으로 지정&해제하는 API. 기본 설정은 수직 방향. / 파라미터 - GenGrid 객체, EINA_TRUE or EINA_FALSE.

Elm_Gengrid_Item_Class *elm_gengrid_item_class_new(void) : GenGrid 항목 클래스를 생성하는 API.

char *app_get_resource_path(void) : /res 폴더의 절대 경로를 반환하는 API.

Elm_Object_Item *elm_gengrid_item_append(Evas_Object *obj, const Elm_Gengrid_Item_Class *gic, const void *data, Evas_Smart_Cb func, const void *func_data) : GenGrid에 새로운 항목을 추가하는 API. / 파라미터는

- GenGrid 객체, 항목 클래스, 항목 데이터 구조체, 항목 선택 이벤트
콜백 함수명, 사용자 데이터.

Evas_Object *elm_image_add(Evas_Object *parent) : 이미지 객체를
생성하는 API.

Eina_Bool elm_image_file_set(Evas_Object *obj, const char *file, const
char *group) : 이미지 객체에 파일 경로를 지정해서 이미지를 로딩하는
API.

void elm_image_aspect_fixed_set(Evas_Object *obj, Eina_Bool
fixed) : Image 객체에서 이미지 잘림 속성을 지정하는 API. 2번째
파라미터에 EINA_TRUE를 전달하면 이미지에 잘리는 부분이 없도록
축소됩니다. EINA_FALSE를 전달하면 항목 영역에 이미지를 꽉 채우기
위해서 잘리는 영역이 발생합니다. 기본 설정은 EINA_TRUE.

18. WebView 위젯으로 만드는 간단한 웹브라우저

웹페이지를 화면에 표시하려면 WebView 위젯을 사용하면 됩니다. WebView는 Evas를 보모로 지정해서 생성해야 합니다. Evas는 EFL에서 사용하는 캔버스입니다. 이번 시간에는 WebView 위젯을 사용해서 간단한 웹 브라우저 예제를 만들어 보겠습니다.

1) WebView 위젯 생성 & 웹페이지 표시

새로운 소스 프로젝트를 생성하고 Project name을 WebViewEx 으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 위부분에 새로운 코드를 추가합니다. 라이브러리 헤더파일을 선언하고 appdata 구조체에 변수를 추가하는 코드입니다.

```
#include "webviewex.h"
#include <EWebKit.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    Evas_Object *entry;
    Evas_Object *web_view;
} appdata_s;
```

create_base_gui() 함수 위에 새로운 함수를 생성합니다. Tabel에 위젯을 추가하는 함수입니다.

```

static void
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)
{
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
    elm_table_pack(table, child, x, y, w, h);
    evas_object_show(child);
}

```

그런 다음 `create_base_gui()` 함수로 이동해서 새로운 코드를 추가합니다. Box, Table, Entry, 3개의 Button, 1개의 WebView 위젯을 생성하는 코드입니다. Conformant와 Label은 이번 예제에서 필요없으니 주석 처리합니다.

```

/* Conformant */
/*ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);*/

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "Hello EFL");
evas_object_size_hint_weight_set(ad->label,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);
evas_object_show(ad->label);*/

```

```

{

/* Box to put the table in so we can bottom-align the table
 * window will stretch all resize object content to win size */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, box);
evas_object_show(box);

/* Table */
Evas_Object *table = elm_table_add(ad->win);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 5 * elm_config_scale_get(), 10 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fiill width but align to bottom of box */
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_box_pack_end(box, table);
evas_object_show(table);

{
/* Entry */
ad->entry = elm_entry_add(ad->win);
elm_entry_scrollable_set(ad->entry, EINA_TRUE);
eext_entry_selection_back_event_allow_set(ad->entry, EINA_TRUE);
elm_object_text_set(ad->entry, "http://www.tizen.org");
my_table_pack(table, ad->entry, 0, 0, 3, 1);

/* Button-1 */
Evas_Object *btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Prev");

```

```

evas_object_smart_callback_add(btn, "clicked", btn_prev_cb, ad);
my_table_pack(table, btn, 0, 1, 1, 1);

/* Button-2 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Go");
evas_object_smart_callback_add(btn, "clicked", btn_go_cb, ad);
my_table_pack(table, btn, 1, 1, 1, 1);

/* Button-3 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Next");
evas_object_smart_callback_add(btn, "clicked", btn_next_cb, ad);
my_table_pack(table, btn, 2, 1, 1, 1);

/* WebView */
Evas *evas = evas_object_evas_get(ad->win);
ad->web_view = ewk_view_add(evas);
ewk_view_url_set(ad->web_view, elm_object_text_get(ad->entry) );
my_table_pack(table, ad->web_view, 0, 2, 3, 8);

}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Conformant와 Label을 주석 처리하고, 1개의 Entry와 3개의 Button, 그리고 1개의 WebView 위젯을 생성하였습니다.

`elm_win_indicator_mode_set()` 함수의 2번째 파라미터에 `ELM_WIN_INDICATOR_SHOW`를 전달하면 화면 위쪽에 인디케이터가 표시되고, `ELM_WIN_INDICATOR_HIDE`를 전달하면 사라집니다. 공간을 최대한 활용하기 위해서는 인디케이터를 감추는 것이 좋습니다.

`evas_object_evas_get()` 은 Evas 객체를 생성하는 API입니다. Evas 는 이미지나 도형을 그릴수 있는 캔버스입니다.

`ewk_view_add()` 은 WebView 위젯을 생성하는 API입니다. WebView 는 부모를 Evas로 지정해야 합니다.

`ewk_view_url_set()` 는 WebView 위젯에 URL 경로를 지정하는 API입니다. 1번째 파라미터에는 WebView 위젯 객체를, 2번째 파라미터에는 URL 경로를 지정합니다. Entry 위젯의 캡션 텍스트를 전달하였는데, <http://www.tizen.org>는 타이젠 개발자 지원 사이트 입니다.

Button을 3개 생성하였으니 콜백 함수도 3개가 필요합니다.

`create_base_gui()` 함수 위에 새로운 함수 3개를 추가합니다. 함수 내용은 잠시 추가해 보겠습니다.

```
static void
btn_go_cb(void *data, Evas_Object *obj, void *event_info)
{
}

static void
btn_prev_cb(void *data, Evas_Object *obj, void *event_info)
{
}

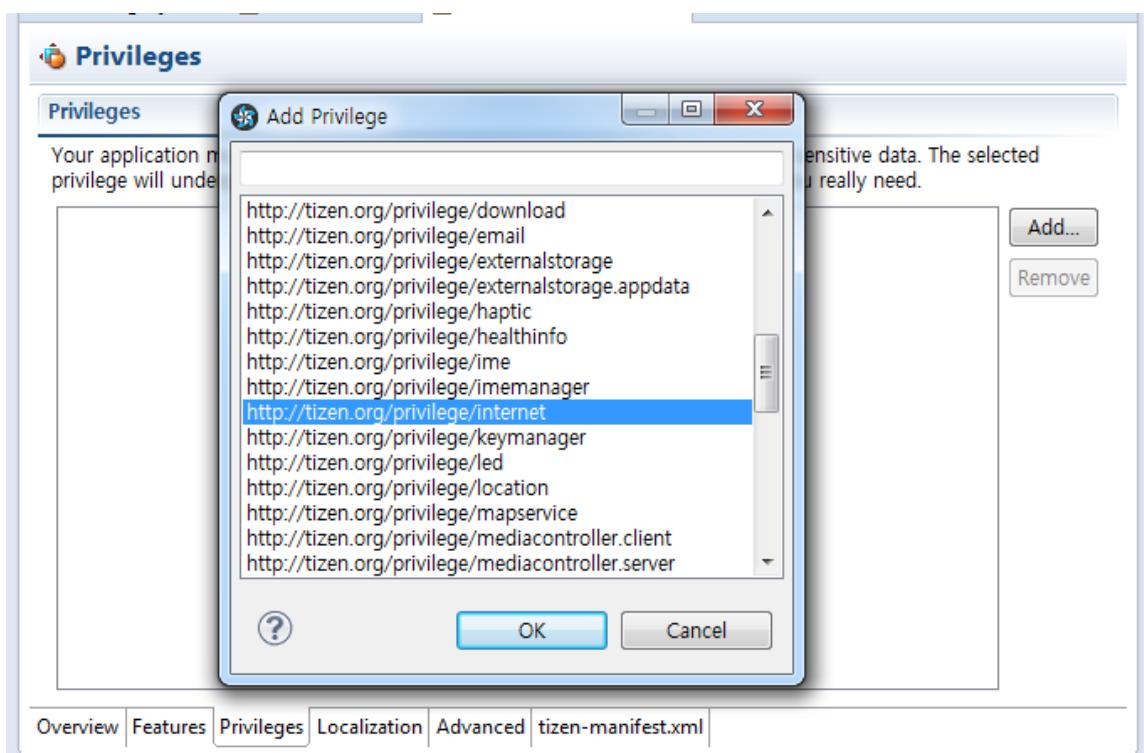
static void
btn_next_cb(void *data, Evas_Object *obj, void *event_info)
{}
```

이 상태에서 실행하면 웹페이지가 표시되지 않습니다. 이유는

네트워크를 사용하기 위해서 사용자 권한(Privilege)을 지원해 줘야하기 때문입니다.

소스 프로젝트 루트 폴더에 있는 tizen-manifest.xml 파일을 더블클릭해서 엽니다. 아래쪽에 여러개의 템버튼이 보이는데 그 중에서 Privileges를 선택합니다.

그런 다음 Add 버튼을 누르고 팝업창이 나타나면 목록에서 http://tizen.org/privilege/internet를 선택하고 OK 버튼을 누릅니다.

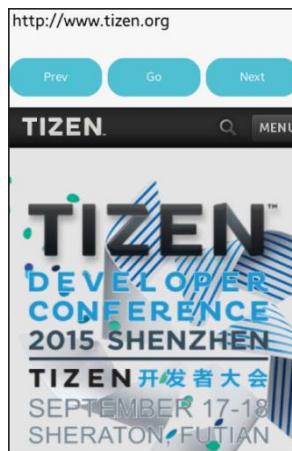


아래쪽 템버튼 중에서 tizen-manifest.xml을 선택하면 소스코드가 나타납니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest xmlns="http://tizen.org/ns/packages" api-version="2.3"
```

```
package="org.example.webviewex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.webviewex" exec="webviewex" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>webviewex</label>
        <icon>webviewex.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/internet</privilege>
    </privileges>
</manifest>
```

이제 이 예제는 네트워크 통신을 사용할 수 있습니다. 예제를 빌드하고 실행시켜 봅시다. 타이젠 개발자 지원 사이트가 보입니다.



2) 웹사이트 이동

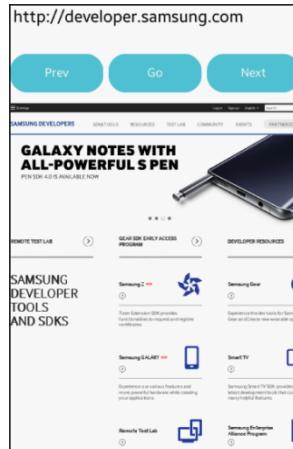
사용자가 Entry 위젯에 URL 주소를 입력하고 Go 버튼을 누르면 해당 주소로 이동하는 기능을 구현해 봅시다. Go 버튼 콜백 함수에 새로운 코드를 추가합니다.

```
static void  
btn_go_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s* ad = data;  
    ewk_view_url_set(ad->web_view, elm_object_text_get(ad->entry) );  
}
```

콜백 함수 1번째 파라미터에는 사용자 데이터가 전달됩니다.

추가된 코드는 Entry 위젯에 입력된 내용을 WebView 위젯에 URL로 지정하는 것입니다.

예제를 다시 실행시켜 봅시다. Entry에 자신이 원하는 웹사이트 주소를 입력하고 Go 버튼을 눌러봅시다. WebView에 해당 웹페이지가 표시됩니다.



3) Prev & Next 페이지로 이동

Prev 버튼을 누르면 이전 페이지로 이동하고, Next 버튼을 누르면 다음

페이지로 이동하는 기능을 구현해 봅시다.

btn_prev_cb() 함수에 새로운 코드를 추가합니다.

```
static void  
btn_prev_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s* ad = data;  
    if( ewk_view_back_possible( ad->web_view ) == EINA_TRUE )  
        ewk_view_back( ad->web_view );  
}
```

ewk_view_back_possible() 는 이전 화면으로 이동 가능한지 여부를 판단하는 API입니다.

ewk_view_back() 는 이전 화면으로 이동하는 API입니다.

그런 다음 btn_next_cb() 함수에 새로운 코드를 추가합니다.

```
static void  
btn_next_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s* ad = data;  
    if( ewk_view_forward_possible( ad->web_view ) == EINA_TRUE )  
        ewk_view_forward( ad->web_view );  
}
```

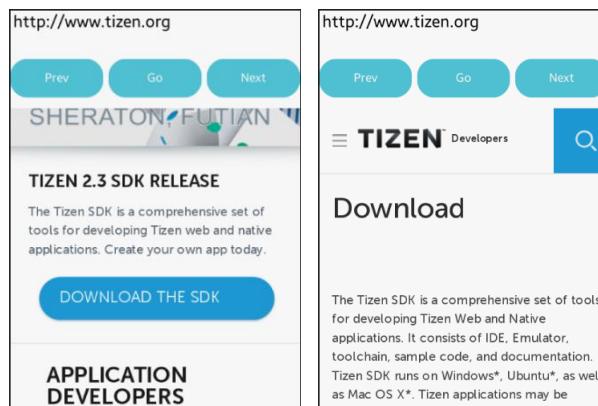
ewk_view_forward_possible() 는 다음 화면으로 이동 가능한지 여부를 판단하는 API입니다.

`ewk_view_forward()` 는 다음 화면으로 이동하는 API입니다.

예제를 다시 실행시켜 봅시다. 웹페이지를 위로 슬라이드하면 'DOWNLOAD THE SDK' 이라는 링크가 나옵니다. 이것을 클릭합니다. 이것이 보이지 않으면 아무 링크나 상관 없습니다.

새로운 웹페이지가 보이면 Prev 버튼을 누릅니다. 첫 화면으로 되돌아오게 됩니다.

이번에는 Next 버튼을 누릅니다. 다시 Download 화면이 나타납니다.



5) 관련 API

Evas *evas_object_evas_get(const Evas_Object *obj) : Evas 객체를 생성하는 API. Evas는 이미지나 도형을 그릴수 있는 캔버스 입니다.

Evas_Object* ewk_view_add(Evas* e) : WebView 위젯을 생성하는 API. WebView는 부모를 Evas로 지정해야 합니다.

Eina_Bool ewk_view_url_set(Evas_Object* o, const char* url) : WebView 위젯에 URL 경로를 지정하는 API. 파라미터 - WebView 위젯 객체, URL 경로.

Eina_Bool ewk_view_back_possible(Evas_Object* o) : 이전 화면으로 이동 가능한지 여부를 판단하는 API.

Eina_Bool ewk_view_back(Evas_Object* o) : 이전 화면으로 이동하는 API.

Eina_Bool ewk_view_forward_possible(Evas_Object* o) : 다음 화면으로 이동 가능한지 여부를 판단하는 API입니다.

Eina_Bool ewk_view_forward(Evas_Object* o) : 다음 화면으로 이동하는 API.

19. Layout으로 구현하는 탭화면

탭화면을 사용하면 간편하게 화면을 전환할 수 있습니다. 이번 예제에서는 Layout과 Box 컨테이너를 사용해서 탭화면을 구현하는 방법을 알아보겠습니다.

1) Layout 컨테이너 생성 & 위젯 배치

새로운 소스 프로젝트를 생성하고 Project name을 LayoutEx으로 지정합니다.

소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 위부분에 새로운 코드를 추가합니다. appdata 구조체에 변수를 추가하는 코드입니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
Evas_Object *box1;
Evas_Object *box2;
} appdata_s;
```

이번 예제에서는 탭화면을 구현해 볼 것입니다. box1은 1번째 탭화면이고, box2는 2번째 탭화면이 됩니다.

create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}

```

그런 다음 `create_base_gui()` 함수로 이동해서 Box, Table, Button을 생성합니다. Conformat와 Label은 주석처리 합니다.

```

/* Conformant */
/*ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);*/

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);*/

{

/* Main Box */
Evas_Object *box = create_box(ad->win);

/* Table */
Evas_Object *table = elm_table_add(ad->win);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 5 * elm_config_scale_get(), 5 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fiill width but align to bottom of box */
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, 1.0);
elm_box_pack_end(box, table);
evas_object_show(table);

```

```

{
    /* Tab Button-1 */
    Evas_Object *btn = elm_button_add(ad->win);
    elm_object_text_set(btn, "Tab-1");
    evas_object_smart_callback_add(btn, "clicked", btn_tab1_cb, ad);
    my_table_pack(table, btn, 0, 5, 1, 1);

    /* Tab Button-2 */
    btn = elm_button_add(ad->win);
    elm_object_text_set(btn, "Tab-2");
    evas_object_smart_callback_add(btn, "clicked", btn_tab2_cb, ad);
    my_table_pack(table, btn, 1, 5, 1, 1);

}
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Table을 생성하였고, 2개의 Button을 Table에 추가하였습니다. Box는 위젯 간의 간격을 지정하기 위해서 사용되었습니다.

Button의 콜백 함수를 생성하겠습니다.

create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다. 함수의 내용은 잠시 후에 만들어 보겠습니다.

```

static void
btn_tab1_cb(void *data, Evas_Object *obj, void *event_info)
{
}

static void

```

```
btn_tab2_cb(void *data, Evas_Object *obj, void *event_info)
{
}
```

예제를 빌드하고 실행시켜 봅시다. 화면 아래쪽에 2개의 Button이 보입니다.



2) 탭화면 생성

2개의 탭화면을 생성해 봅시다. `create_base_gui()` 함수에 새로운 코드를 추가합니다. 2개의 Layout과 Box를 생성하고, 그 위에 위젯을 추가하는 코드입니다.

```
/* Tab Button-2 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Tab-2");
evas_object_smart_callback_add(btn, "clicked", btn_tab2_cb, ad);
my_table_pack(table, btn, 1, 5, 1, 1);

/* Layout-1 */
Evas_Object *layout1 = elm_layout_add(ad->win);
elm_layout_theme_set(layout1, "layout", "drawer", "panel");
my_table_pack(table, layout1, 0, 0, 2, 5);

/* Box-1 */
ad->box1 = create_box(layout1);
elm_win_resize_object_add(ad->win, ad->box1);

{
    /* Label */
    ad->label = elm_label_add(layout1);
    elm_object_text_set(ad->label, "Tab-1");
    my_box_pack(ad->box1, ad->label, 1.0, 0.0, -1.0, 0.5);
}

/* Layout-2 */
Evas_Object *layout2 = elm_layout_add(ad->win);
elm_layout_theme_set(layout2, "layout", "drawer", "panel");
my_table_pack(table, layout2, 0, 0, 2, 5);
```

```

/* Box-2 */
ad->box2 = create_box(layout2);
elm_win_resize_object_add(ad->win, ad->box2);
evas_object_hide(ad->box2);

{
    /* Button */
    btn = elm_button_add(layout2);
    elm_object_text_set(btn, "Tab-2");
    my_box_pack(ad->box2, btn, 1.0, 0.0, -1.0, 0.5);
}
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

2개의 Layout을 생성하였고 그 위에 Box를 추가하였습니다. Layout 이 템화면 역할을 하게 됩니다. 1번째 템화면에는 Label을 추가하였고, 2번째 템화면에는 Button을 추가하였습니다.

`elm_layout_add()` 은 Layout 컨테이너를 생성하는 API입니다.

`elm_layout_theme_set()` 은 Layout 테마 스타일을 지정하는 API입니다. Layout 안에 윈도우를 배치하려면 테마 스타일을 panel로 지정해야 합니다. 방법은 아래와 같이 파라미터를 전달하면 됩니다.

`elm_layout_theme_set(layout, "layout", "drawer", "panel")`

`evas_object_hide()` 는 윈도우를 감추는 API입니다. `evas_object_show()` 와 반대 기능입니다.

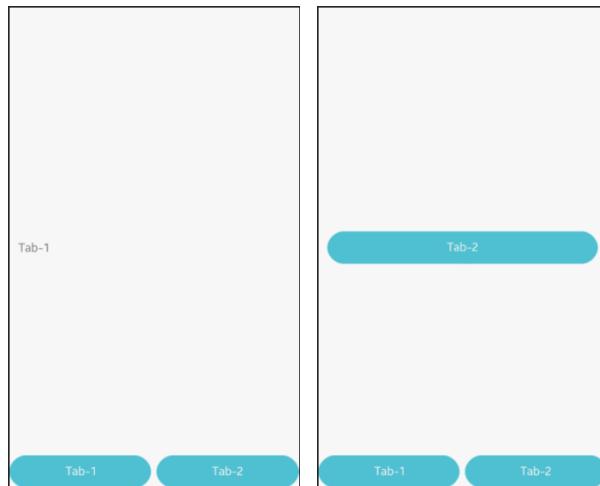
화면 아래쪽에 있는 2개의 Button을 누르면 해당되는 탭화면을 표시하고, 나머지는 감추는 기능을 구현해 보겠습니다. Button 콜백 함수에 새로운 코드를 추가합니다.

```
static void  
btn_tab1_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    evas_object_show(ad->box1);  
    evas_object_hide(ad->box2);  
}  
  
static void  
btn_tab2_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    evas_object_hide(ad->box1);  
    evas_object_show(ad->box2);  
}
```

1번째 Button을 누르면 1번째 Layout을 보여주고 2번째 Layout은 감춥니다.

2번째 Button을 누르면 1번째 Layout을 감추고 2번째 Layout은 보여줍니다.

예제를 다시 실행시켜 봅시다. 화면 아래쪽 Button을 누르면 해당되는 탭화면이 나타납니다.



4) 관련 API

Evas_Object *elm_layout_add(Evas_Object *parent) : Layout 컨테이너를 생성하는 API.

Eina_Bool elm_layout_theme_set(Evas_Object *obj, const char *clas, const char *group, const char *style) : Layout 테마 스타일을 지정하는 API. Layout 안에 윈도우를 배치하려면 테마 스타일을 panel로 지정해야 합니다.

void evas_object_hide(Evas_Object *obj) : 윈도우를 감추는 API.
evas_object_show() 와 반대 기능.

20. Naviframe으로 구현하는 헤더 & 네비게이션바

Naviframe를 사용하면 Header와 Footer(Toolbar)를 구현할 수 있습니다. 헤더에 타이틀 텍스트를 표시하고 풋터에 네비게이션바를 표시하는 방법을 알아보겠습니다.

1) Header 생성

새로운 소스 프로젝트를 생성하고 Project name을 NaviframeEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 위부분에 새로운 코드를 추가합니다. appdata 구조체에 변수를 추가하는 코드입니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
Evas_Object *nf;
Evas_Object *layout;
Elm_Object_Item *frame_item;
Evas_Object *toolbar;
Elm_Object_Item *btn1;
Elm_Object_Item *btn2;
Elm_Object_Item *btn3;
} appdata_s;
```

nf는 Naviframe 객체이고, layout은 Naviframe에 추가되는 메인 컨테이너입니다.

Elm_Object_Item 는 Naviframe의 Item 구조체이다.

btn1 ~ btn3은 네비게이션바에 추가되는 Button 입니다.

create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",  
     * "pad_large" and "pad_huge" available as styles in addition to the  
     * "default" frame style */  
    elm_object_style_set(frame, "pad_medium");  
    /* set the input weight/aling on the frame insted of the child */  
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);  
    evas_object_size_hint_align_set(frame, h_align, v_align);  
    {  
        /* tell the child that is packed into the frame to be able to expand */  
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,  
                                         EVAS_HINT_EXPAND);  
        /* fill the expanded area (above) as opposaed to center in it */  
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);  
        /* actually put the child in the frame and show it */  
        evas_object_show(child);  
        elm_object_content_set(frame, child);  
    }  
    /* put the frame into the box instead of the child directly */  
    elm_box_pack_end(box, frame);  
    /* show the frame */
```

```
    evas_object_show(frame);
}

-----
```

그런 다음 `create_base_gui()` 함수로 이동해서 새로운 코드를 추가합니다. `Naviframe`, `Layout`, `Naviframe Item`을 생성하는 코드입니다.

```
-----  
/* Conformant */  
ad->conform = elm_conformant_add(ad->win);  
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);  
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);  
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
elm_win_resize_object_add(ad->win, ad->conform);  
evas_object_show(ad->conform);  
  
{  
/* Naviframe */  
ad->nf = elm_naviframe_add(ad->conform);  
elm_object_part_content_set(ad->conform, "elm.swallow.content", ad->nf);  
elm_object_content_set(ad->conform, ad->nf);  
  
/* child object - indent to how relationship */  
/* A box to put things in vertically - default mode for box */  
Evas_Object *box = elm_box_add(ad->conform);  
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
elm_object_content_set(ad->nf, box);  
evas_object_show(box);  
  
{  
/* Label*/  
ad->label = elm_label_add(ad->conform);  
elm_object_text_set(ad->label, "Press Toolbar Button");
```

```

my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.0);

/* Header */
ad->frame_item = elm_naviframe_item_push(ad->nf, "Naviframe Ex",
NULL, NULL, box, NULL);

/* Toolbar */
ad->toolbar = toolbar_add(ad, ad->nf);
elm_object_item_part_content_set(ad->frame_item, "toolbar", ad-
>toolbar);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Conformant 위에 Naviframe을 추가하였고, 다시 그 위에 Box를 추가하였습니다.

`elm_naviframe_add()` 은 Naviframe 객체를 생성하는 API입니다.

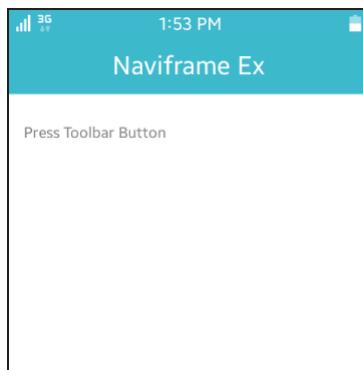
`evas_object_size_hint_weight_set()` 는 객체의 크기에 대한 힌트를 지정하는 API입니다. 파라미터는 순서대로 객체, 수평 크기 힌트, 수직 크기 힌트입니다. EVAS_HINT_EXPAND 최대한 크게 지정하는 옵션입니다.

`evas_object_size_hint_align_set()` 는 객체의 정렬 방식에 대한 힌트를 지정하는 API입니다. 파라미터는 순서대로 객체, 수평 정렬 방식 힌트, 수직 정렬 방식 힌트입니다. EVAS_HINT_FILL 는 주어진 공간에 최대한 채우는 옵션입니다.

`elm_naviframe_item_push()` 는 Naviframe Item을 생성하는 API입니다. 파라미터는 순서대로 Naviframe 객체, 타이틀 텍스트, 이전 Item 으로

이동하는 Button 객체, 다음 Item으로 이동하는 Button 객체, 콘텐츠, Item 스타일입니다.

예제를 빌드하고 실행시켜 볼시다. 헤더에 타이틀 텍스트가 표시되었습니다.



2) Footer에 툴바 표시

이번에는 Naveframe Item에 Footer를 추가하고 3개의 탭버튼을 생성해 보겠습니다.

create_base_gui() 함수 위에 새로운 함수를 추가합니다. Toolbar를 생성해서 반환하는 함수입니다.

```
static Evas_Object *toolbar_add(appdata_s *ad, Evas_Object *parent)
{
    Evas_Object *toolbar = elm_toolbar_add(parent);
    evas_object_show(toolbar);

    ad->btn1 = elm_toolbar_item_append(toolbar, NULL, "Left", on_btn1_cb, ad);
    ad->btn2 = elm_toolbar_item_append(toolbar, NULL, "Center", on_btn2_cb, ad);
    ad->btn3 = elm_toolbar_item_append(toolbar, NULL, "Right", on_btn3_cb, ad);
```

```
    return toolbar;  
}  
_____
```

elm_toolbar_add() 는 Toolbar 객체를 생성하는 API입니다.

elm_toolbar_item_append() 는 Toolbar에 항목을 추가하는 API입니다.
파라미터는 순서대로 Toolbar 객체, 아이콘 이미지, 캡션 텍스트, Button
클릭 이벤트 콜백 함수명, 사용자 데이터 입니다.

탭버튼의 콜백 함수를 생성해 봅시다. toolbar_add() 함수 위에 새로운
함수 3개를 추가합니다. 사용자가 탭버튼을 클릭하면 Label 위젯에
텍스트를 변경하는 코드입니다.

```
_____  
static void on_btn1_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = (appdata_s*)data;  
    elm_object_text_set(ad->label, "Button-1 Pressed");  
}
```

```
static void on_btn2_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = (appdata_s*)data;  
    elm_object_text_set(ad->label, "Button-2 Pressed");  
}
```

```
static void on_btn3_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = (appdata_s*)data;  
    elm_object_text_set(ad->label, "Button-3 Pressed");  
}
```

이제 `elm_toolbar_add()` 함수를 호출해서 Toolbar를 생성하면 됩니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Header */
ad->frame_item = elm_naviframe_item_push(ad->nf, "Naviframe Ex", NULL,
NULL, box, NULL);

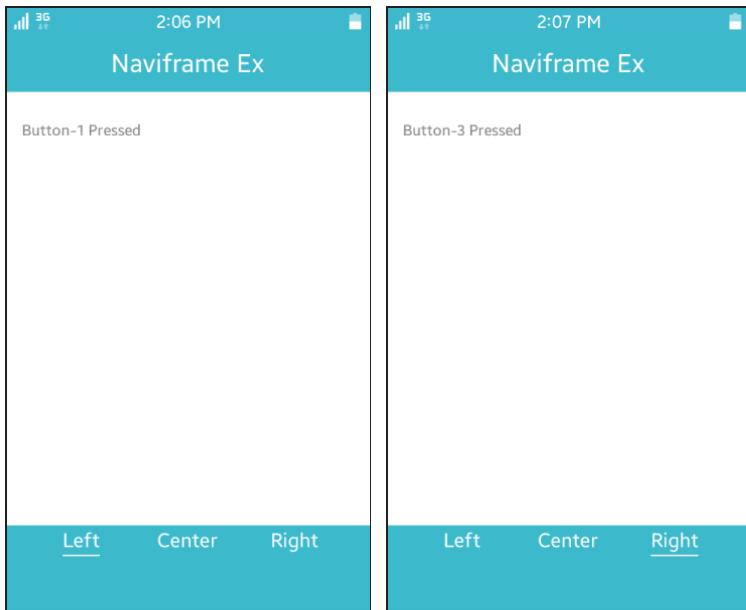
/* Toolbar */
ad->toolbar = toolbar_add(ad, ad->nf);
elm_object_item_part_content_set(ad->frame_item, "toolbar", ad->toolbar);
}

}
```

`toolbar_add()` 함수를 호출해서 Toolbar를 생성하였습니다. 그런 다음 Toolbar를 Naveframe Item의 콘텐츠로 지정합니다.

`elm_object_item_part_content_set()` 은 객체 Item에 콘텐츠를 지정하는 API입니다. 파라미터는 순서대로 객체 Item, 콘텐츠 Part 이름, 콘텐츠 객체입니다.

예제를 다시 실행시켜 봅시다. 화면 아래쪽에 Footer가 보이고 그 안에 3개의 Button이 있습니다. Button을 클릭하면 Label 위젯에 텍스트가 변경됩니다. 현재 선택된 Button에는 선택표시가 나타납니다.



3) Toolbar 속성 변경

텍스트가 위쪽에 위치해 있어서 보기 좋지 않습니다. 마지막으로 텍스트 위치를 중앙으로 이동하고, 선택 표시를 생략하도록 하겠습니다.

`toolbar_add()` 함수에 새로운 코드 2줄을 추가합니다.

```

static Evas_Object *toolbar_add(appdata_s *ad, Evas_Object *parent)
{
    Evas_Object *toolbar = elm_toolbar_add(parent);
    evas_object_show(toolbar);
    elm_toolbar_select_mode_set(toolbar, ELM_OBJECT_SELECT_MODE_NONE);
    elm_toolbar_transverse_expanded_set(toolbar, EINA_TRUE);

    ad->btn1 = elm_toolbar_item_append(toolbar, NULL, "Left", on_btn1_cb, ad);
    ad->btn2 = elm_toolbar_item_append(toolbar, NULL, "Center", on_btn2_cb, ad);
    ad->btn3 = elm_toolbar_item_append(toolbar, NULL, "Right", on_btn3_cb, ad);
}

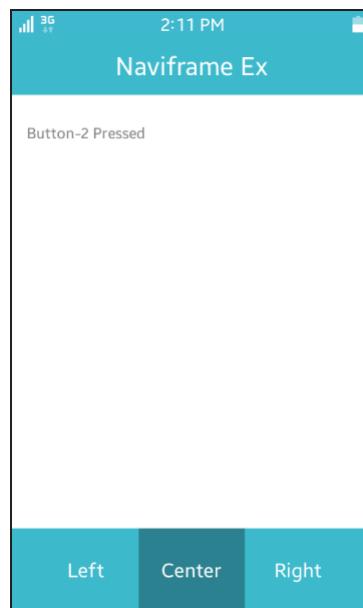
```

```
    return toolbar;  
}  
_____
```

elm_toolbar_select_mode_set() 는 Toolbar의 선택 모드를 지정하는 API입니다. 2번째 파라미터에 ELM_OBJECT_SELECT_MODE_NONE를 전달하면 선택 표시가 사라집니다. 기본 설정은 ELM_OBJECT_SELECT_MODE_DEFAULT 입니다.

elm_toolbar_transverse_expanded_set() 는 Toolbar 크기 확장 여부를 지정하는 API 입니다. 2번째 파라미터에 EINA_TRUE를 전달하면 툴바 크기가 Footer를 가득 채워지기 때문에 텍스트가 중앙에 표시됩니다.

예제를 다시 실행시켜 봅시다. Toolbar의 텍스트가 중앙에 표시되고, 탭버튼을 누르면 선택 표시는 보이지 않습니다.



4) 관련 API

Elm_Object_Item : 객체 Item 혹은 Naviframe의 Item의 구조체.

Elm_Object_Item *elm_naviframe_item_push(Evas_Object *obj, char *title_label, Evas_Object *prev_btn, Evas_Object *next_btn, Evas_Object *content, char *item_style) : Naviframe 객체를 생성하는 API. / 파라미터 - Naviframe 객체, 타이틀 텍스트, 이전 Item 으로 이동하는 Button 객체, 다음 Item 으로 이동하는 Button 객체, 콘텐츠, Item 스타일.

void evas_object_size_hint_weight_set(Evas_Object *obj, double x, double y) : 객체의 크기에 대한 힌트를 지정하는 API. / 파라미터 - 객체, 수평 크기 힌트, 수직 크기 힌트. EVAS_HINT_EXPAND 는 최대한 크게 지정하는 옵션.

void evas_object_size_hint_align_set(Evas_Object *obj, double x, double y) : 객체의 정렬 방식에 대한 힌트를 지정하는 API. / 파라미터 - 객체, 수평 정렬 방식 힌트, 수직 정렬 방식 힌트. EVAS_HINT_FILL은 주어진 공간에 최대한 채우는 옵션.

Elm_Object_Item *elm_naviframe_item_push(Evas_Object *obj, char *title_label, Evas_Object *prev_btn, Evas_Object *next_btn, Evas_Object *content, char *item_style) : Naviframe Item을 생성하는 API. / 파라미터 - Naviframe 객체, 타이틀 텍스트, 이전 Item 으로 이동하는 Button 객체, 다음 Item 으로 이동하는 Button 객체, 콘텐츠, Item 스타일.

Evas_Object *elm_toolbar_add(Evas_Object *parent) : Toolbar를 생성하는 API.

Elm_Object_Item *elm_toolbar_item_append(Evas_Object *obj, char *icon, char *label, Evas_Smart_Cb func, void *data) : Toolbar에 항목을 추가하는

API. / 파라미터 - Toolbar 객체, 아이콘 이미지, 캡션 텍스트, Button 클릭 이벤트 콜백 함수명, 사용자 데이터.

void elm_object_item_part_content_set(Elm_Object_Item *it, char *part, Evas_Object *content) : 객체 Item에 콘텐츠를 지정하는 API. /
파라미터 : 객체 Item, 콘텐츠 Part 이름, 콘텐츠 객체.

void elm_toolbar_select_mode_set(Evas_Object *obj, Elm_Object_Select_Mode mode) : Toolbar의 선택 모드를 지정하는 API. /
파라미터 - Toolbar 객체, 선택 모드, 기본 설정은
ELM_OBJECT_SELECT_MODE_DEFAULT.

ELM_OBJECT_SELECT_MODE_NONE를 전달하면 선택 표시가 사라집니다.

void elm_toolbar_transverse_expanded_set(Evas_Object *obj, Eina_Bool transverse_expanded) : Toolbar 크기 확장 여부를 지정하는 API. /
파라미터 - Toolbar 객체, 크기 확장 여부. EINA_TRUE를 전달하면 툴바 크기가 Footer를 가득 채워지기 때문에 텍스트가 중앙에 표시됩니다.

21. Box 컨테이너로 위젯을 순차적으로 배치하기

해상도가 다른 여러 단말을 지원하려면 위젯을 배치할 때 상대좌표로 배치하면 편리합니다. Table 컨테이너의 경우에는 비율 단위로 위젯의 좌표를 지정할 수 있으며, Box 컨테이너는 순차적으로 위젯을 배치하거나 상대좌표(좌,우,가운데 or 위,아래,중앙)로 배치할 때 사용합니다. 안드로이드의 LinearLayout 과 유사한 기능입니다. 이번 예제에서는 Box 컨테이너를 사용해서 위젯을 수평방향과 수직방향으로 배치하는 방법을 알아보겠습니다.

1) 수평 Box 생성

이번 예제에서는 3개의 Box 컨테이너와 6개의 Button 위젯을 생성하게 됩니다. 소스코드를 간결하게 하기 위해서 Box와 Button 생성 함수를 먼저 만들어 보겠습니다.

새로운 소스 프로젝트를 생성하고 Project name을 BoxEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

```
static Evas_Object*
create_box(Evas_Object *parent)
{
    Evas_Object *box = elm_box_add(parent);
    evas_object_show(box);
    return box;
}

static Evas_Object *
```

```
create_button(Evas_Object *parent, char *text)
{
    Evas_Object *button = elm_button_add(parent);
    elm_object_text_set(button, text);
    evas_object_show(button);
    return button;
}
```

create_box() 는 Box 컨테이너를 생성해서 반환하는 함수입니다.

elm_box_add() 는 Box 컨테이너를 생성하는 API 입니다.

create_button() Button 위젯을 생성하고 캡션 텍스트를 지정해서 반환하는 함수입니다.

방금 만든 함수를 이용해서 2개의 Box와 3개의 Button을 생성한 다음 수평 방향으로 배치해 보겠습니다. create_base_gui() 함수로 이동해서 새로운 코드를 추가합니다. Label은 이번 예제에서 필요없으니 삭제 합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = create_box(ad->win);
```

```

elm_box_padding_set(box, 10, 10);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);

{ /* child object - indent to how relationship */
/* Create horizontal box */
Evas_Object *horizontal_box = create_box(ad->win);
elm_box_horizontal_set(horizontal_box, EINA_TRUE);
elm_box_padding_set(horizontal_box, 10, 10);
elm_box_pack_end(box, horizontal_box);

{ /* child object - indent to how relationship */
Evas_Object *btn = create_button(horizontal_box, "Left");
elm_box_pack_end(horizontal_box, btn);

btn = create_button(horizontal_box, "Mid");
evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0.0);
evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.0);
elm_box_pack_end(horizontal_box, btn);

btn = create_button(horizontal_box, "Right");
elm_box_pack_end(horizontal_box, btn);
}

}
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

elm_box_padding_set() 은 Box 컨테이너의 안쪽 여백을 지정하는 API입니다. 파라미터는 순서대로 컨테이너 객체, 왼쪽 & 오른쪽 여백, 위쪽 & 아래쪽 여백입니다.

`evas_object_size_hint_weight_set()` 는 객체가 차지하는 영역의 크기에 대한 힌트를 지정하는 API입니다. 파라미터는 순서대로 객체, 수평 크기 힌트, 수직 크기 힌트입니다. `EVAS_HINT_EXPAND` 는 최대한 크게 지정하는 옵션입니다.

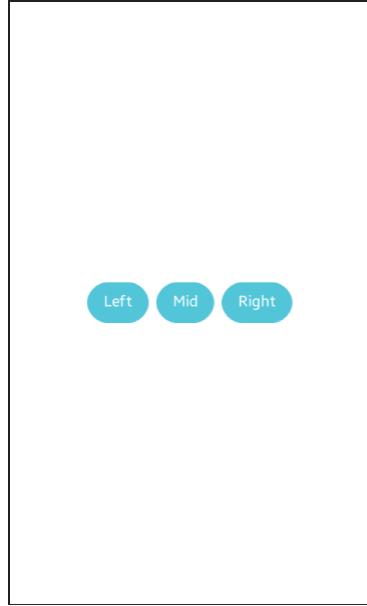
`evas_object_size_hint_align_set()` 는 객체 자체의 크기에 대한 힌트를 지정하는 API입니다. 파라미터는 순서대로 객체, 수평 크기 힌트, 수직 크기 힌트입니다. `EVAS_HINT_FILL` 는 최대한 크게 지정하는 옵션입니다.

`elm_win_resize_object_add()` 는 특정 객체의 크기를 다른 객체와 동일하게 지정하는 API입니다. 1번째 파라미터에 `win`을 전달했고, 2번째 파라미터에 `main_box`를 전달했습니다. `main_box`의 크기가 화면 전체를 차지하게 되는 것입니다.

`elm_box_horizontal_set()` 는 Box의 배치방향을 지정하는 API입니다. 2번째 파라미터에 `EINA_TRUE`를 전달하면 수평방향이 되고, `EINA_FALSE`를 전달하면 수직방향이 됩니다. 기본 설정은 수직방향입니다.

`elm_box_pack_end()` 는 Box 컨테이너에 새로운 객체를 추가하는 API입니다. 수평방향이라면 오른쪽에 추가되고, 수직방향이라면 아래쪽에 추가됩니다.

예제를 빌드하고 실행시켜 봅시다. 화면 중앙에 3개의 Button이 수평으로 배치되어 있습니다. 분명히 2번째 Button의 크기를 최대로 지정했는데 최소한의 영역만을 차지하고 있습니다. 이유는 Box 컨테이너의 넓이를 지정하지 않았기 때문입니다.



2) 객체 크기 최대화

Box의 넓이를 최대로 지정하는 방법을 알아보겠습니다. `create_base_gui()` 함수의 2번째 Box를 생성하는 코드에 새로운 코드를 추가합니다.

```
/* Create horizontal box */
Evas_Object *horizontal_box = create_box(ad->win);
elm_box_horizontal_set(horizontal_box, EINA_TRUE);
elm_box_padding_set(horizontal_box, 10, 10);
evas_object_size_hint_weight_set(horizontal_box, EVAS_HINT_EXPAND, 0.0);
evas_object_size_hint_align_set(horizontal_box, EVAS_HINT_FILL, 0.0);
elm_box_pack_end(main_box, horizontal_box);
```

`evas_object_size_hint_weight_set()` 함수의 2번째 파라미터에 `EVAS_HINT_EXPAND`를 전달해서 차지하는 넓이를 최대로 지정하였습니다. 3번째 파라미터에는 0.0을 지정해서 차지하는 높이를

최소로 지정하였습니다.

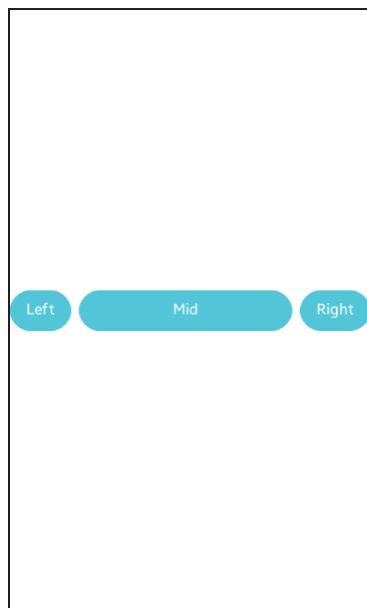
이것만으로는 Button의 넓이가 변경되지 않습니다.

evas_object_size_hint_weight_set() 는 영역을 확보하는 기능을 담당하고, 컨텐츠의 정렬방식을 지정하는 함수는 evas_object_size_hint_align_set() 사용해야 합니다.

evas_object_size_hint_align_set() 함수의 2번째 파라미터는 수평 정렬방식을 지정합니다. 0.0은 왼쪽 정렬, 0.5는 중앙 정렬, 1.0은 오른쪽 정렬을 의미하고 EVAS_HINT_FILL을 전달하면 컨텐츠의 넓이를 최대로 지정합니다.

3번째 파라미터는 수직 정렬방식을 지정합니다. 0.0은 위쪽 정렬, 0.5는 가운데 정렬, 1.0은 아래쪽 정렬을 의미하고 EVAS_HINT_FILL을 전달하면 컨텐츠의 높이를 최대로 지정합니다.

다시 실행시켜 봅시다. 2번째 Button의 넓이가 확장되었습니다.



3) Box 위치 변경

Button이 중앙에 위치해 있는 이유는 2번째 Box가 중앙에 위치해 있기 때문입니다. Box 컨테이너는 순차적으로 컨텐츠를 배치하기 때문에 1번째 Box에 새로운 Box를 추가하면 2번째 Box는 위로 올라가게 됩니다. `create_base_gui()` 함수 아래 부분에 새로운 코드를 추가합니다. 3번째 Box를 생성해서 1번째 Box에 추가하는 코드입니다.

```
    }
}

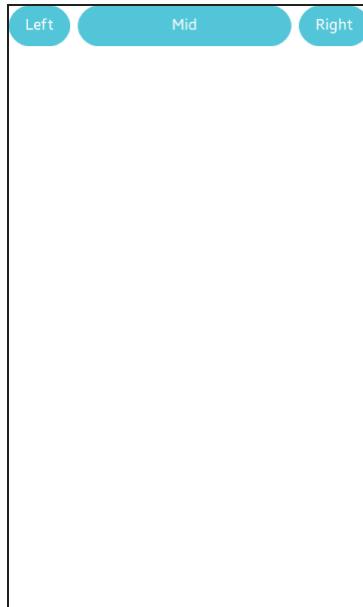
/* Create vertical box */
Evas_Object *vertical_box = create_box(ad->win);
elm_box_padding_set(vertical_box, 10, 10);
// Set area size
evas_object_size_hint_weight_set(vertical_box,      EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(vertical_box,      EVAS_HINT_FILL,
EVAS_HINT_FILL);
elm_box_pack_end(box, vertical_box);
}
}
```

1번째 Box의 배치 방향은 수직이기 때문에 새로 추가된 Box는 2번째 Box의 아래쪽에 위치하게 됩니다.

`evas_object_size_hint_weight_set()` 함수에 `EVAS_HINT_EXPAND`를 전달해서 영역을 최대로 확보했고, `evas_object_size_hint_align_set()` 함수에 `EVAS_HINT_FILL`를 전달해서 내부 컨텐츠의 크기도 최대로 지정하였습니다.

예제를 다시 실행해 보면 Button이 위쪽으로 이동한 것을 확인할수

있습니다.



4) 수직방향으로 위젯 배치

3번째 Box에 3개의 Button을 추가해 보겠습니다. 배치방향을 별도로 지정하지 않았기 때문에 Button은 수직방향으로 배치됩니다.

create_base_gui() 함수 아래 부분에 새로운 코드를 추가합니다.

```
/* Create vertical box */
Evas_Object *vertical_box = create_box(ad->win);
elm_box_padding_set(vertical_box, 10, 10);
// Set area size
evas_object_size_hint_weight_set(vertical_box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(vertical_box, EVAS_HINT_FILL,
EVAS_HINT_FILL);
elm_box_pack_end(box, vertical_box);
```

```

{ /* child object - indent to how relationship */
    Evas_Object *btn = create_button(vertical_box, "Top");
    evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0.0);
    evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.0);
    elm_box_pack_end(vertical_box, btn);

    btn = create_button(vertical_box, "Center");
    evas_object_size_hint_weight_set(btn,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(btn,           EVAS_HINT_FILL,
EVAS_HINT_FILL);
    elm_box_pack_end(vertical_box, btn);

    btn = create_button(vertical_box, "Bottom");
    evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0.0);
    evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.0);
    elm_box_pack_end(vertical_box, btn);
}

}
}

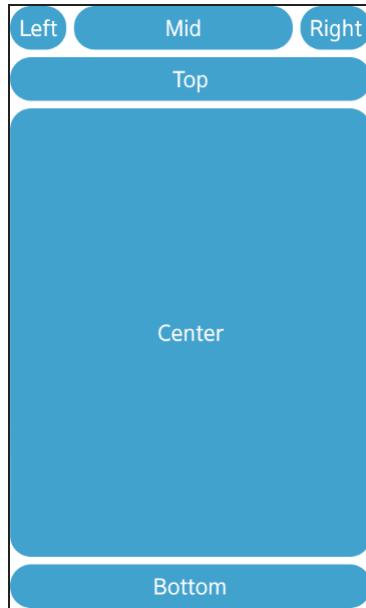
```



1번째와 3번째 Button은 넓이를 최대, 높이는 최소로 지정했습니다.

2번째 Button은 넓이와 높이를 모두 최대로 지정했습니다.

예제를 다시 실행해 봅시다. 새로 추가된 3개의 Button이 수직방향으로 배치되어 있습니다.



5) 관련 API

`Evas_Object *elm_box_add(Evas_Object *parent)` : Box 컨테이너를 생성하는 API.

`void elm_box_padding_set(Evas_Object *obj, Evas_Coord horizontal, Evas_Coord vertical)` : 안쪽 여백을 지정하는 API. / 파라미터 - 컨테이너 객체, 왼쪽 & 오른쪽 여백, 위쪽 & 아래쪽 여백.

`void evas_object_size_hint_weight_set(Evas_Object *obj, double x, double y)` : 객체가 차지하는 영역의 크기에 대한 힌트를 지정하는 API. / 파라미터 - 객체, 수평 크기 힌트, 수직 크기 힌트. `EVAS_HINT_EXPAND` 는 최대한 크게 지정하는 옵션.

`void evas_object_size_hint_align_set(Evas_Object *obj, double x, double y)` : 객체 자체의 크기에 대한 힌트를 지정하는 API. / 파라미터 - 객체, 수평 크기 힌트, 수직 크기 힌트. `EVAS_HINT_FILL` 는 최대한 크게

지정하는 옵션.

void elm_win_resize_object_add(Evas_Object *obj, Evas_Object *subobj) : 특정 객체의 크기를 다른 객체와 동일하게 지정하는 API. /
파라미터 - 윈도우 객체, 크기 변경되는 객체.

void elm_box_horizontal_set(Evas_Object *obj, Eina_Bool horizontal) : box의 배치방향을 지정하는 API. / 파라미터
- 객체, 배치 방향. EINA_TRUE를 전달하면 수평방향이 되고,
EINA_FALSE를 전달하면 수직방향이 됩니다. 기본 설정은 수직방향.

void elm_box_pack_end(Evas_Object *obj, Evas_Object *subobj) : box 컨테이너에 새로운 객체를 추가하는 API. 수평방향이라면 오른쪽에
추가되고, 수직방향이라면 아래쪽에 추가됩니다. / 파라미터 - Box 객체,
컨텐츠 객체.

22. Scroller로 Sub 페이지 만들기

상용 앱을 개발할 때는 여러개의 페이지를 제작하게 됩니다. 멀티 페이지를 구현할 때는 다음과 같은 방법이 있습니다.

- Toolbar와 여러개의 layout을 이용해서 탭버튼을 눌렀을 때 layout을 show/hide 해주는 방법.
- Scroller로 layout을 이동하는 방법.

이번 시간에는 Scroller로 페이지를 이동하는 방법을 알아보겠습니다.

2번째 페이지에 해당하는 소스파일도 생성하겠습니다.

1) 메인 페이지 UI 작업

메인 페이지에 Naviframe을 이용해서 헤더를 표시하고, 1개의 버튼을 추가해 보겠습니다.

새로운 소스 프로젝트를 생성하고 Project name을 Multipage 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 appdata 구조체에 변수를 하나 추가합니다. 제목을 표시하는 Naviframe 객체를 위한 변수입니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *nf;  
    Evas_Object *label;  
} appdata_s;
```

`create_base_gui()` 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}
```

그런 다음 create_base_gui() 함수로 이동해서 아래와 같이 코드를 수정합니다. Naviframe, Box, Label, Button, Header를 생성하는 코드입니다.

```
//eext_object_event_callback_add(ad->win, EEXT_CALLBACK_BACK, win_back_cb,
ad);

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* Naviframe */
ad->nf = elm_naviframe_add(ad->conform);
elm_object_part_content_set(ad->conform, "elm.swallow.content", ad->nf);
elm_object_content_set(ad->conform, ad->nf);

Evas_Object *box = elm_box_add(ad->nf);
elm_box_padding_set(box, 10 * elm_config_scale_get(), 10 *
elm_config_scale_get());
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->nf, box);
evas_object_show(box);

{

/* Label*/
ad->label = elm_label_add(ad->conform);
```

```

elm_object_text_set(ad->label, "Press Button");
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.0);

/* Button */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Sub Window");
my_box_pack(box, btn, 1.0, 0.0, -1.0, 0.0);

/* Header */
Elm_Object_Item *nf_it;
nf_it = elm_naviframe_item_push(ad->nf, "Main Window", NULL, NULL,
box, NULL);
eext_object_event_callback_add(ad->nf, EEXT_CALLBACK_BACK,
eext_naviframe_back_cb, NULL);
elm_naviframe_item_pop_cb_set(nf_it, naviframe_pop_cb, ad->win);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

아래 코드는 Back 버튼을 눌렀을 때 앱이 종료되는 기능을 합니다.
문제는 Sub 페이지에서 Back 버튼을 눌렀을 때도 종료된다는 것입니다.
그래서 이 코드는 일단 주석 처리합니다.

```
eext_object_event_callback_add(ad->win, EEXT_CALLBACK_BACK,
win_back_cb, ad);
```

아래는 Sub 페이지에서 Back 버튼을 눌렀을 때 Main 페이지로 되돌아
오는 코드입니다.

```
eext_object_event_callback_add(ad->nf, EEXT_CALLBACK_BACK,  
eext_naviframe_back_cb, NULL);
```

Main 페이지에서 Back 버튼을 눌렀을 때 앱이 종료되는 기능도 필요합니다. win_back_cb 콜백 함수를 호출하는 코드를 주석처리 했기 때문입니다. 그래서 아래 코드를 추가한 것입니다. Main 페이지에서 Back 버튼을 누르면 naviframe_pop_cb 함수가 호출됩니다.

```
elm_naviframe_item_pop_cb_set(nf_it, naviframe_pop_cb, ad->win);
```

이제 naviframe_pop_cb 콜백 함수를 만들어 봅시다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static Eina_Bool  
naviframe_pop_cb(void *data, Elm_Object_Item *it)  
{  
    ui_app_exit();  
    return EINA_FALSE;  
}
```

Main 페이지에서 Back 버튼을 누르면 위 함수가 호출됩니다.

ui_app_exit() 는 앱을 종료하는 함수입니다.

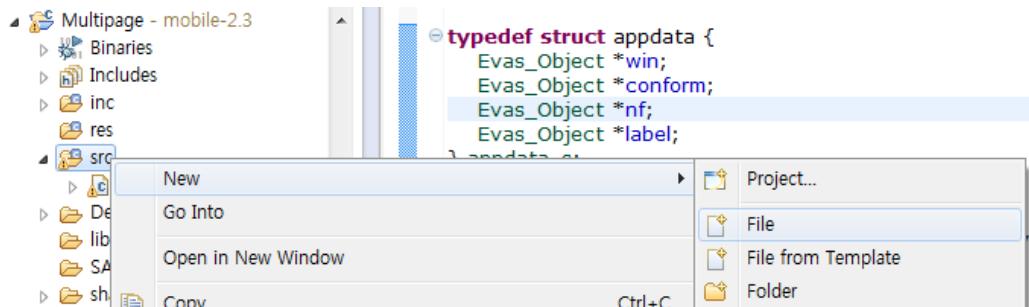
예제를 빌드하고 실행시켜 봅시다. Header와 Label, 그리고 Button이 보입니다.



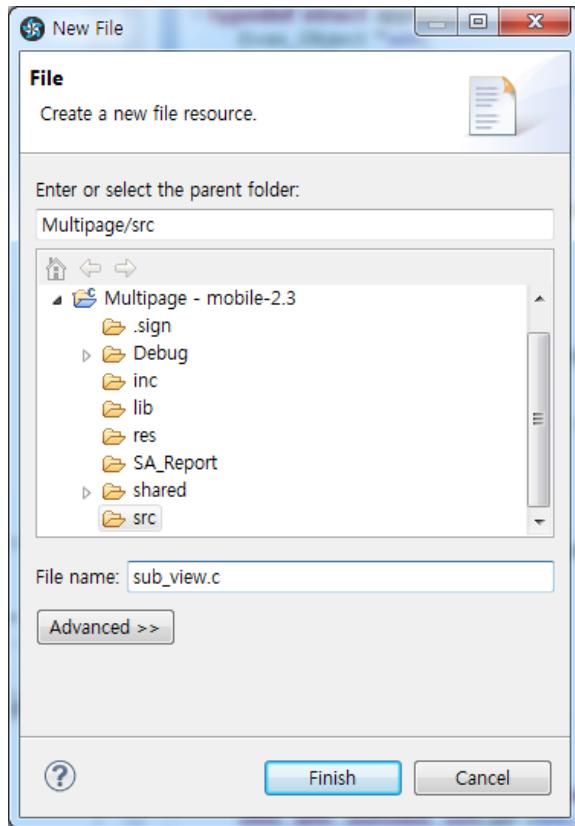
2) Sub 페이지 생성

Button을 클릭하면 Sub 페이지가 나타나는 기능을 구현해 봅시다. 메인 소스파일(multipage.c)에서 Sub 페이지 생성 코드를 추가해도 상관은 없습니다. 하지만 코드가 길어지면 관리하기 힘들어 집니다. 파일을 분할하면 다음번 앱을 개발할 때 다시 사용하기도 수월합니다.

새로운 소스파일을 생성하겠습니다. /src 폴더를 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [New > File]을 선택합니다.



팝업창이 나타나면 File Name 항목에 sub_view.c 이라고 입력하고 Finish 버튼을 눌러서 닫습니다. 그러면 /src 폴더에 새로운 파일이 생성됩니다.



새로 생성된 소스파일(sub_view.c)을 더블클릭해서 편집 모드로 열어줍니다. Sub 페이지와 아이콘 버튼을 생성하는 코드를 추가해 보겠습니다.

```
#include "multipage.h"

static Evas_Object*
create_button_view(Evas_Object *parent)
{
    Evas_Object *btn, *img, *box;

    box = elm_box_add(parent);
```

```

evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(box, EVAS_HINT_FILL, EVAS_HINT_FILL);

/* icon_reorder style */
btn = elm_button_add(box);
elm_object_style_set(btn, "icon_reorder");
evas_object_show(btn);
elm_box_pack_end(box, btn);

return box;
}

void
sub_view_cb(void *data, Evas_Object *obj, void *event_info)
{
    Evas_Object *scroller, *layout;
    Evas_Object *nf = data;

    scroller = elm_scroller_add(nf);
    layout = create_button_view(scroller);
    elm_object_content_set(scroller, layout);

    elm_naviframe_item_push(nf, "Sub Window", NULL, NULL, scroller, NULL);
}

```

1번째 페이지와 연동해야 하기 때문에 메인 헤더파일(multipage.h)을 선언합니다.

`create_button_view()` 는 Box 컨테이너를 생성하고 그 위에 Button 위젯을 추가하는 함수입니다.

`sub_view_cb()` 는 Scroller와 Layout을 생성해서 2번째 페이지를 화면에 표시하는 함수입니다.

`elm_scroller_add()` 는 새로운 Scroller를 생성하는 API입니다.

`elm_object_content_set()` 는 컨테이너의 콘텐츠를 지정하는 API입니다.
1번째 파라미터에 Scroller를 지정하고, 2번째 파라미터에 layout을
지정하면 layout 이 화면에 보이는 것입니다.

`elm_naviframe_item_push()` 함수를 이용해서 Header 타이틀 텍스트를
변경하고 Scroller를 Naviframe의 콘텐츠로 지정합니다.

1번째 페이지의 Button을 클릭했을 때 `sub_view_cb()` 함수를 호출해주면
됩니다. 그러기 위해서 헤더파일에 함수를 선언해 주어야 합니다. /inc
폴더를 열고 `multipage.h` 파일을 더블클릭 합니다. 파일이 열리면 아래
부분에 Sub 페이지 생성 함수를 선언해 줍니다.

```
_____
#ifndef PACKAGE
#define PACKAGE "org.example.multipage"
#endif

void sub_view_cb(void *data, Evas_Object *obj, void *event_info);

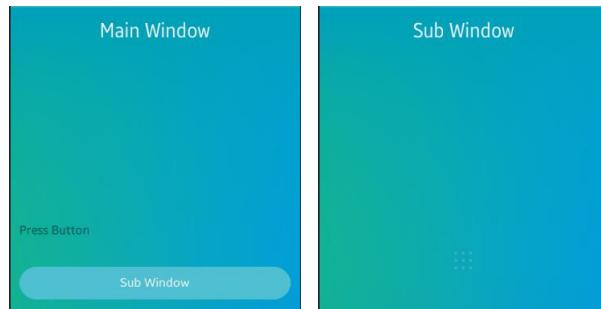
#endif /* __multipage_H__ */
_____
```

이제 Main 페이지에서 위 함수를 호출할 수 있습니다. `Multipage.c`
파일로 이동해서 `create_base_gui()` 함수의 Button 생성 코드에 한줄을
추가합니다. Main 페이지의 Button을 클릭했을 때 Sub 페이지를 생성
함수를 호출하는 코드입니다.

```
_____
/* Button */
```

```
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Sub Window");
evas_object_smart_callback_add(btn, "clicked", sub_view_cb, ad->nf);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 0.0);
```

예제를 다시 실행시켜 봅시다. Button을 누르면 Sub 페이지가 나타나고 중앙에 아이콘 버튼이 보입니다. Main 페이지로 돌아가고 싶으면 Back 버튼을 누르면 됩니다.



3) Button 클릭으로 Main 페이지로 이동

Sub 페이지에 있는 Button을 클릭하면 Main 페이지로 이동하는 기능을 구현해 보겠습니다. 방법은 `elm_naviframe_item_pop()` 함수를 호출해 주면 됩니다.

`sub_view.c` 파일의 `sub_view_cb()` 함수로 이동해서 코드를 수정해 줍니다. `create_button_view()` 함수에 Naviframe을 전달하는 것입니다.

```
scroller = elm_scroller_add(nf);
layout = create_button_view(scroller, nf);
//layout = create_button_view(scroller);
elm_object_content_set(scroller, layout);
```

```
    elm_naviframe_item_push(nf, "Sub Window", NULL, NULL, scroller, NULL);  
}  
}
```

그런 다음 `create_button_view()` 함수에 새로운 코드를 추가합니다.

```
static Evas_Object*  
create_button_view(Evas_Object *parent, Evas_Object *nf)  
{  
    Evas_Object *btn, *img, *box;  
  
    box = elm_box_add(parent);  
    evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);  
    evas_object_size_hint_align_set(box, EVAS_HINT_FILL, EVAS_HINT_FILL);  
  
    /* icon_reorder style */  
    btn = elm_button_add(box);  
    elm_object_style_set(btn, "icon_reorder");  
    evas_object_smart_callback_add(btn, "clicked", btn_back_cb, nf);  
    evas_object_show(btn);  
~
```

함수 파라미터에 NaviFrame 객체가 전달됩니다. Button 콜백 함수를 `btn_back_cb` 으로 지정했고, 사용자 데이터로 Naviframe을 전달하였습니다.

마지막으로 `create_button_view()` 함수 위에 Button 콜백 함수를 추가합니다.

```
void  
btn_back_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    Evas_Object *nf = data;  
    elm_naviframe_item_pop(nf);  
}
```

`elm_naviframe_item_pop()` 은 Naviframe의 스택에 쌓여있는 페이지 목록 중에서 가장 위쪽으로 이동하는 명령어입니다. Main 페이지로 이동하는 것입니다.

다시 예제를 실행하고 Sub 페이지로 이동했다가 아이콘 Button을 클릭해 봅시다. Main 페이지로 이동합니다.

Sub 페이지에서 Back 버튼을 누르면 Main 페이지로 이동하고, Main 페이지에서 Back 버튼을 누르면 앱화면이 사라집니다. 종료된 것은 아니고 Background 모드로 전환된 것입니다.

4) 관련 API

`eext_object_event_callback_add(ad->win, EEXT_CALLBACK_BACK, win_back_cb, ad)` : Back 버튼을 눌렀을 때 앱을 종료하는 코드입니다.
Sub 페이지에서 Back 버튼을 눌렀을 때 종료되지 않으려면 이 코드를 주석 처리해야합니다.

`eext_object_event_callback_add(ad->nf, EEXT_CALLBACK_BACK, eext_naviframe_back_cb, NULL)` : Sub 페이지에서 Back 버튼을 눌렀을 때 Main 페이지로 되돌아 오는 코드입니다.

`elm_naviframe_item_pop_cb_set(nf_it, naviframe_pop_cb, ad->win)` : Main 페이지에서 Back 버튼을 눌렀을 때의 콜백 함수를 naviframe_pop_cb 으로 지정하는 코드입니다.

`ui_app_exit()` : 앱을 종료하는 API.

`Evas_Object *elm_naviframe_add(Evas_Object *parent)` : 새로운 Scroller를 생성하는 API.

`void elm_object_content_set(Evas_Object *obj, Evas_Object *content)` : 컨테이너의 콘텐츠를 지정하는 API. / 파라미터 - 컨테이너 객체, 콘텐츠 객체

`Elm_Object_Item *elm_naviframe_item_push(Evas_Object *obj, const char *title_label, Evas_Object *prev_btn, Evas_Object *next_btn, Evas_Object *content, const char *item_style)` : Naviframe의 Header 타이틀 텍스트와 콘텐츠를 지정하는 API. / 파라미터 - Naviframe 객체, 타이틀 텍스트, 이전 Button 객체, 다음 Button 객체, 콘텐츠, 사용자 데이터

`Evas_Object *elm_naviframe_item_pop(Evas_Object *obj)` : Naviframe의 스택에 쌓여있는 페이지 목록 중에서 가장 위쪽으로 이동하는 API. / 파라미터 - Naviframe 객체.

23. 문자열 사용방법

문자열 변환 혹은 검색은 앱을 개발할 때 자주 사용되는 기능입니다. 이번 예제에서는 C 언어 기본 API를 사용해서 문자열을 복사하고, 특정 문자열의 위치를 검색하고, 문자열을 숫자로 변경하는 방법을 알아보겠습니다.

1) 문자열 복사

새로운 소스 프로젝트를 생성하고 Project name을 StringEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수로 이동해서 새로운 코드를 추가합니다.

```
/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "Hello EFL");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);
elm_label_line_wrap_set(ad->label, EINA_TRUE);

/* Show window after base gui is set up */
evas_object_show(ad->win);

show_string_result(ad->label);
```

elm_label_line_wrap_set() 함수는 Label 위젯에 자동 줄바꿈을 지정하는 API 입니다.

`show_string_result()` 는 문자열 변환 결과를 화면에 출력하는 함수입니다. 이제부터 만들어 보겠습니다.

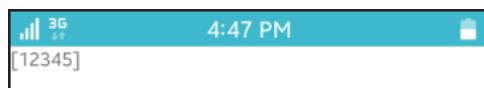
`create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void  
show_string_result(Evas_Object *label)  
{  
    char buf[PATH_MAX], str1[100];  
    char *str2;  
    strcpy(str1, "12345");  
    sprintf(buf, "[%s]", str1);  
  
    elm_object_text_set(label, buf);  
}
```

`strcpy()` 는 문자열을 복사하는 API입니다. 1번째 파라미터는 문자열이 채워지는 메모리 주소, 2번째는 원본 문자열 데이터입니다. 여기서는 `str1`이라는 문자열 변수에 '12345' 라는 글자를 저장하였습니다.

`sprintf()` 는 포맷 형식을 지정해서 새로운 문자열을 생성하는 API입니다. 1번째 파라미터는 문자열이 채워지는 메모리 주소, 2번째는 포맷 형식, 3번째 부터는 포맷에 대입되는 데이터입니다.

예제를 빌드하고 실행시켜 봅시다. 원본 문자열을 대괄호로 묶어서 출력하였습니다.



2) 문자열 길이 구하기

str1 변수에 저장된 문자열의 길이를 구해보겠습니다.

show_string_result() 함수 끝부분에 새로운 코드를 추가합니다.

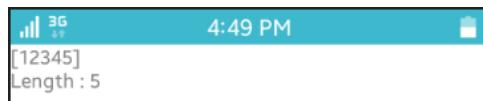
```
strcpy(str1, "12345");
sprintf(buf, "[%s]", str1);

int length = strlen(str1);
sprintf(buf, "%s<br>Length : %d", buf, length);

elm_object_text_set(label, buf);
```

strlen() 은 문자열의 길이를 구해서 반환하는 API입니다. str1은 char 배열이라서 길이가 고정되어 있습니다. 이런 경우에는 End 기호(₩0) 바로 앞까지의 길이를 반환하게 됩니다.

예제를 다시 실행시켜 봅시다. 문자열의 길이가 표시됩니다.



3) 길이를 지정해서 문자열 앞부분을 추출

str1 변수에서 앞부분 3글자를 추출해 보겠습니다. show_string_result() 함수 끝부분에 새로운 코드를 추가합니다.

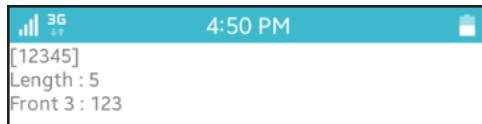
```
sprintf(buf, "%s<br>Length : %d", buf, length);
```

```
str2 = eina_stringshare_add_length(str1, 3);
sprintf(buf, "%s<br>Front 3 : %s", buf, str2);

elm_object_text_set(label, buf);
```

eina_stringshare_add_length() 는 길이를 지정해서 문자열의 앞부분을 추출하는 API입니다. 1번째 파라미터는 원본 문자열 데이터이고, 2번째는 추출할 길이입니다. 추출된 문자열을 반환합니다.

예제를 다시 실행시켜 봅시다. '12345'에서 앞부분 '123'이 추출되었습니다.



4) 문자열 일부분을 추출

시작 위치를 지정해서 문자열의 일정 길이 만큼 추출하는 방법을 알아보겠습니다. show_string_result() 함수 끝부분에 새로운 코드를 추가합니다.

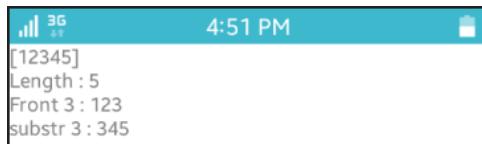
```
sprintf(buf, "%s<br>Front 3 : %s", buf, str2);

str2 = eina_stringshare_add_length(str1 + 2, 3);
sprintf(buf, "%s<br>substr 3 : %s", buf, str2);

elm_object_text_set(label, buf);
```

eina_stringshare_add_length() 함수의 1번째 파라미터에 str1 + 2를 전달하였습니다. 이렇게 하면 '345'를 전달하게 됩니다.

예제를 다시 실행시켜 봅시다. '12345'에서 3번째 위치 부터 3자리가 추출되었습니다.



5) 문자를 숫자로 변경

문자를 숫자로 변경하려면 atoi() 함수를 사용하면 됩니다.
show_string_result() 함수 끝부분에 새로운 코드를 추가합니다.

```
 sprintf(buf, "%s<br>substr 3 : %s", buf, str2);

int i = atoi(str1);
sprintf(buf, "%s<br>string to int '%s' + 3 = %d", buf, str1, i + 3);

elm_object_text_set(label, buf);
```

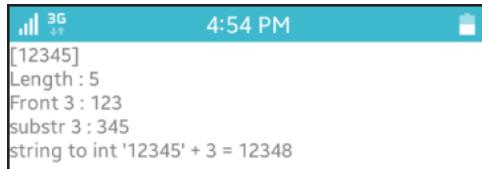
atoi() 는 문자를 int 타입으로 변경하는 API입니다.

atol() 는 문자를 long 타입으로 변경하는 API입니다.

atof 는 문자를 float 타입으로 변경하는 API입니다.

예제를 다시 실행시켜 봅시다. 문자 '12345' 가 숫자로

변경되어서 3을 더한 결과가 출력 되었습니다.



6) 숫자를 문자로 변경

숫자를 문자로 변경하려면 sprintf()를 사용하면 간편합니다.
show_string_result() 함수 끝부분에 새로운 코드를 추가합니다.

```
sprintf(buf, "%s<br>string to int '%s' : %d", buf, str1, i);

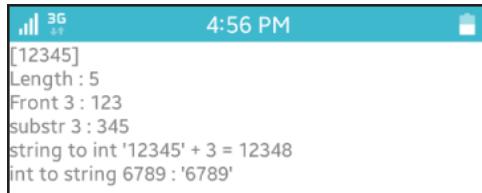
char str3[100];
i = 6789;
sprintf(str3, "%d", i);
sprintf(buf, "%s<br>int to string %d : '%s'", buf, i, str3);

elm_object_text_set(label, buf);
```

문자열 포맷문에서 주로 사용되는 기호는 다음과 같습니다.

- %s 는 문자열을 대입하는 기호입니다.
- %d 는 int 혹은 long 같은 숫자를 대입하는 기호입니다.
- %c 는 char 문자 1개를 대입하는 기호입니다.
- %f 는 float 혹은 double 같은 실수를 대입하는 기호입니다.

예제를 다시 실행시켜 봅시다. 숫자 6789가 문자로 변경되었습니다.



7) 문자열 합치기

2개의 문자열을 하나로 합치는 방법을 알아보겠습니다.
show_string_result() 함수 끝부분에 새로운 코드를 추가합니다.

```
sprintf(buf, "%s<br>int to string %d : '%s'", buf, i, str3);

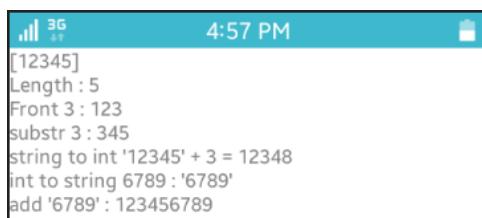
strcat(str1, str3);

sprintf(buf, "%s<br>add '%s' : %s", buf, str3, str1);
```

```
elm_object_text_set(label, buf);
```

strcat() 는 2개의 문자열을 합치는 함수입니다. 1번째 파라미터의 문자열 끝에 2번째 문자열을 덧붙입니다. 새로운 문자열을 생성하는 것이 아니라 1번째 문자열에 추가되는 것입니다

예제를 다시 실행시켜 봅시다. '12345'에 '6789'를 추가해서 '123456789'가 만들어 졌습니다.



8) 문자열 위치 검색

특정 문자열의 위치를 검색하는 방법을 알아보겠습니다.
show_string_result() 함수 끝부분에 새로운 코드를 추가합니다.

```
    sprintf(buf, "%s<br>add '%s' : %s", buf, str3, str1);

str1[0] = '\0';
str3[0] = '\0';

strcpy(str1, "This is a simple string");
sprintf(buf, "%s<br><br>[%s]", buf, str1);

elm_object_text_set(label, buf);
str2 = strstr( str1, "simple" );
sprintf(buf, "%s<br>search 'simple' : %s", buf, str2);

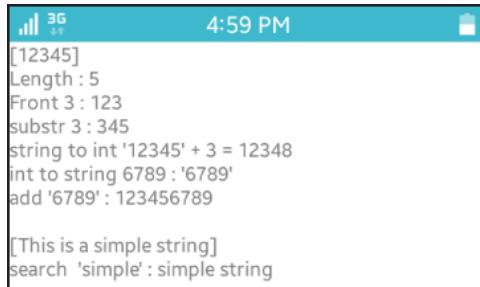
elm_object_text_set(label, buf);
```

'\0' 는 문자열의 끝을 의미하는 End 기호입니다. char 배열의 1번째에 이 기호를 저장하면 문자열이 빈칸으로 초기화 되는 것입니다.

strstr() 는 문자열에 포함된 특정 문자열의 위치를 검색하는 API입니다. 인덱스 번호를 반환하는 것이 아니라 해당 문자열이 시작되는 포인터를 반환합니다. 따라서 문자열이 반환되는 것입니다.

문자열에서 특정 char 1개의 시작 위치를 검색할 때는 strchr(char*, int) 함수를 사용하면 됩니다.

예제를 다시 실행시켜 봅시다. 'simple'이라는 문자가 시작되는 위치 이후의 문자열이 표시됩니다.



9) 길이를 지정해서 문자열 복사

'simple'이라는 글자를 'sample'로 변경해 보겠습니다. `strncpy()` 함수를 사용하면 됩니다. `show_string_result()` 함수 끝부분에 새로운 코드를 추가합니다.

```
sprintf(buf, "%s<br>search 'simple' : %s", buf, str2);

strncpy( str2, "sample", 6 );
sprintf(buf, "%s<br>change 'simple' to 'sample' : %s", buf, str1);

elm_object_text_set(label, buf);
```

`strncpy()`는 길이를 지정해서 문자열을 복사하는 함수입니다. `str2`는 'simple'이라는 글자가 시작되는 위치를 가리키기 때문에 결론적으로 'simple'이 'sample'로 변경되는 것입니다.

예제를 다시 실행시켜 봅시다. 'simple'이 'sample'로 변경되었습니다.

The screenshot shows a mobile application interface with a teal header bar. The top left corner displays signal strength and battery level. The top center shows the time as 5:01 PM. The main content area contains several lines of text demonstrating string operations:

```
[12345]
Length : 5
Front 3 : 123
substr 3 : 345
string to int '12345' + 3 = 12348
int to string 6789 : '6789'
add '6789' : 123456789

[This is a simple string]
search 'simple' : simple string
change 'simple' to 'sample' : This is a sample string
```

10) 관련 API

`char *strcpy (char *dest, char *src)` : 문자열을 복사하는 API. / 파라미터 - 문자열이 채워지는 메모리 주소, 원본 문자열 데이터.

`int sprintf (char *s, char *format, ...)` : 포맷 형식을 지정해서 새로운 문자열을 생성하는 API. / 파라미터 : 문자열이 채워지는 메모리 주소, 포맷 형식, 3번째 부터는 포맷에 대입되는 데이터.

`size_t strlen (char *s)` : 문자열의 길이를 구해서 반환하는 API. 문자열 포맷문에서 `%s` 는 문자열을 대입하는 기호. `%d` 는 `int` 혹은 `long` 같은 숫자를 대입하는 기호. `%c` 는 `char` 문자 1개를 대입하는 기호. `%f` 는 `float` 혹은 `double` 같은 실수를 대입하는 기호.

`Eina_Stringshare *eina_stringshare_add_length(const char *str, unsigned int slen)` : 길이를 지정해서 문자열의 앞부분을 추출하는 API. / 파라미터 : 원본 문자열 데이터, 추출할 길이. / 반환 : 추출된 문자열.

`int atoi (char *nptr)` : 문자를 `int` 타입으로 변경하는 API.

`long int atol (char *nptr)` : 문자를 `long` 타입으로 변경하는 API.

double atof (char *nptr) : 문자를 float 타입으로 변경하는 API.

char *strcat (char *_dest, char *_src) : 2개의 문자열을 합치는 API.

1번째 파라미터의 문자열 끝에 2번째 문자열을 덧붙입니다. 새로운 문자열을 생성하는 것이 아니라 1번째 문자열에 추가됩니다.

char *strstr (char *haystack, char *needle) : 문자열에 포함된 특정 문자열의 위치를 검색하는 API. 인덱스 번호를 반환하는 것이 아니라 해당 문자열이 시작되는 포인터를 반환합니다. 따라서 문자열이 반환됩니다.

char *strchr (char *s, int c) : 문자열에서 특정 char 1개의 시작 위치를 검색하는 API.

char *strncpy (char *dest, char *src, size_t n) : 길이를 지정해서 문자열을 복사하는 API.

24. 문자열 구조체 Eina_Strbuf

C언어 기본 API를 사용해서 문자열 핸들링을 하다보면 한계가 느껴질 때가 있습니다. EFL에서는 Eina_Strbuf라는 문자열 구조체를 제공하고 있습니다. Eina_Strbuf를 사용해서 문자열의 일부분을 삭제하고, 다른 문자열로 변경하고, 중간에 새로운 문자열을 삽입하는 방법을 알아보겠습니다.

1) Eina_Strbuf에 문자열 입력

새로운 소스 프로젝트를 생성하고 Project name을 EinaStrbufEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수로 이동해서 새로운 코드를 추가합니다.

```
/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "Hello EFL");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);
elm_label_line_wrap_set(ad->label, EINA_TRUE);

/* Show window after base gui is set up */
evas_object_show(ad->win);

show_eina_strbuf_result(ad->label);
```

elm_label_line_wrap_set() 함수는 Label 위젯에 자동 줄바꿈을 지정하는

API 입니다.

show_eina_strbuf_result() 는 Eina_Strbuf 사용 결과를 화면에 출력하는 함수입니다. 이제부터 만들어 보겠습니다.

create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void  
show_eina_strbuf_result(Evas_Object *label)  
{  
    Eina_Strbuf *strline, *strbuf;  
    /* Create Eina_Strbuf */  
    strbuf = eina_strbuf_new();  
    /* Addend string */  
    eina_strbuf_append(strbuf, "Hello Tizen");  
  
    elm_object_text_set(label, eina_strbuf_string_get(strbuf));  
    /* Free memory */  
    eina_strbuf_free(strbuf);  
}
```

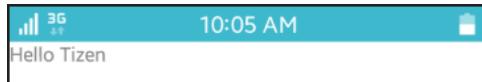
eina_strbuf_new() 는 새로운 Eina_Strbuf 객체를 생성하는 API입니다.

eina_strbuf_append() 는 Eina_Strbuf에 문자열을 추가하는 API입니다.

eina_strbuf_string_get() 는 Eina_Strbuf에 저장된 문자열을 구하는 API입니다.

eina_strbuf_free() 는 Eina_Strbuf에 저장된 데이터를 삭제하는 API입니다.

예제를 빌드하고 실행시켜 봅시다.



2) 포맷을 지정해서 문자열 추가

문자열 포맷을 지정해서 새로운 문자열을 추가할 때는 `eina_strbuf_append_printf()` 함수를 사용하면 됩니다. `show_eina_strbuf_result()` 함수 중간에 새로운 코드를 추가합니다.

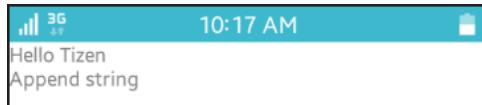
```
/* Reset string */
strline = eina_strbuf_new();
eina_strbuf_append(strline, "Append string");
eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );

elm_object_text_set(label, eina_strbuf_string_get(strbuf));
/* Free memory */
eina_strbuf_free(strline);
eina_strbuf_free(strbuf);
```

`eina_strbuf_new()` 함수를 사용해서 새로운 Eina_Strbuf 객체를 생성했고, `eina_strbuf_append()` 함수를 사용해서 문자열을 입력했습니다.

`eina_strbuf_append_printf()` 는 포맷문을 지정해서 새로운 문자열을 추가하는 API입니다. 1번째 파라미터는 Eina_Strbuf 객체, 2번째는 포맷 형식, 3번째 부터는 포맷에 대입되는 데이터 입니다.

예제를 다시 실행시켜 봅시다. 2줄의 문자열이 출력되었습니다.



3) 문자열 길이 구하기

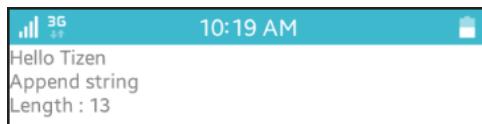
Eina_Strbuf에 저장된 문자열의 길이를 구할 때는 eina_strbuf_length_get() 함수를 사용하면 됩니다. show_eina_strbuf_result() 함수 끝부분에 새로운 코드를 추가합니다.

```
eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );  
  
/* Length of string */  
eina_strbuf_append_printf(strbuf, "<br>Length : %d",  
    eina_strbuf_length_get(strline) );
```

```
elm_object_text_set(label, eina_strbuf_string_get(strbuf));
```

eina_strbuf_length_get() 는 Eina_Strbuf에 저장된 문자열의 길이를 반환하는 API입니다.

예제를 다시 실행시켜 봅시다. 'Append string'의 길이 13이 출력되었습니다.



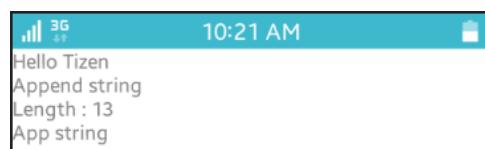
4) 문자열 일부분 삭제

문자열의 일부분만을 삭제할 때는 `eina_strbuf_remove()` 함수를 사용하면 됩니다. `show_eina_strbuf_result()` 함수 끝부분에 새로운 코드를 추가합니다.

```
    eina_strbuf_append_printf(strbuf, "<br>Length : %d", eina_strbuf_length_get(strline) );  
  
/* Remove part of string */  
eina_strbuf_remove(strline, 3, 6);  
eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );  
  
elm_object_text_set(label, eina_strbuf_string_get(strbuf));
```

`eina_strbuf_remove()` 는 문자열의 일부분만을 삭제하는 API입니다. 1번째 파라미터는 `Eina_Srbuf` 객체, 2번째는 삭제 범위 시작 위치, 3번째는 삭제 범위 끝 위치입니다. 6을 지정하면 6번째 문자까지 삭제됩니다.

예제를 다시 실행시켜 봅시다. 인덱스 번호 3번(4번째 문자)부터 5번(6번째 문자)까지 삭제되었습니다.



5) 문자열 치환

특정 문자를 다른 문자로 변경하려면 `eina_strbuf_replace()` 함수를 사용하면 됩니다. `show_eina_strbuf_result()` 함수 끝부분에 새로운 코드를

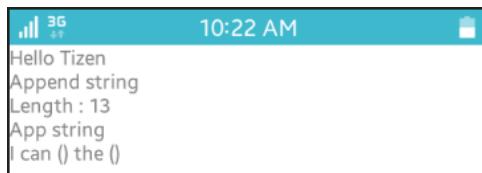
추가합니다.

```
eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );  
  
/* Replace string */  
eina_strbuf_reset(strline);  
eina_strbuf_append(strline, "I () () the ()");  
eina_strbuf_replace(strline, "()", "can", 1);  
eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );  
  
elm_object_text_set(label, eina_strbuf_string_get(strbuf));
```

eina_strbuf_reset() 는 Eina_Strbuf에 저장된 문자열을 초기화 하는 API입니다.

eina_strbuf_replace() 는 특정 문자를 다른 문자로 변경하는 함수입니다. 1번째 파라미터는 Eina_Strbuf 객체, 2번째는 변경할 문자, 3번째는 변경될 문자, 4번째는 변경 횟수 입니다.

예제를 다시 실행시켜 봅시다. 1번째 ')' 가 'can' 으로 변경되었습니다.



6) 모든 문자열 치환

이번에는 동일한 문자를 모두 치환하는 방법을 알아보겠습니다.
show_eina_strbuf_result() 함수 끝부분에 새로운 코드를 추가합니다.

```

eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );

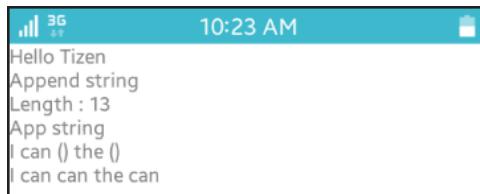
/* Replace all */
eina_strbuf_replace_all(strline, "()", "can");
eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );

elm_object_text_set(label, eina_strbuf_string_get(strbuf));

```

eina_strbuf_replace_all() 는 동일한 문자를 모두 치환하는 API입니다.
 1번째 파라미터는 Eina_Strbuf 객체, 2번째는 변경할 문자, 3번째는
 변경될 문자입니다.

예제를 다시 실행시켜 봅시다. ')' 가 모두 'can' 으로 변경되었습니다.



7) 중간에 문자열 삽입

이번에는 문자열 중간에 문자를 삽입하는 방법을 알아보겠습니다.
 show_eina_strbuf_result() 함수 끝부분에 새로운 코드를 추가합니다.

```

eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );

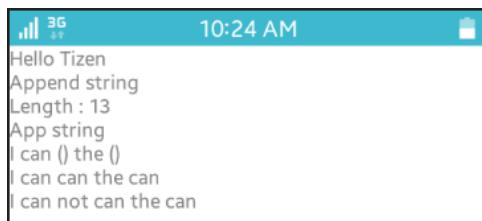
/* Insert string */
eina_strbuf_insert(strline, " not", 5);
eina_strbuf_append_printf(strbuf, "<br>%s", eina_strbuf_string_get(strline) );

```

```
elm_object_text_set(label, eina_strbuf_string_get(strbuf));
```

eina_strbuf_insert() 는 문자열의 특정 위치에 문자를 삽입하는 API입니다. 1번째 파라미터는 Eina_Strbuf 객체, 2번째는 삽입할 문자열, 3번째는 삽입 위치입니다.

예제를 다시 실행시켜 봅시다. 인덱스 번호 5번(6번째)에 'not' 이 삽입되었습니다.



8) 관련 API

Eina_Strbuf *eina_strbuf_new(void) : 새로운 Eina_Strbuf 객체를 생성하는 API.

Eina_Bool eina_strbuf_append(Eina_Strbuf *buf, char *str) : Eina_Strbuf에 문자열을 추가하는 API.

char *eina_strbuf_string_get(Eina_Strbuf *buf) : Eina_Strbuf에 저장된 문자열을 구하는 API.

void eina_strbuf_free(Eina_Strbuf *buf) : Eina_Strbuf에 저장된 데이터를 삭제하는 API.

Eina_Bool eina_strbuf_append_printf(Eina_Strbuf *buf, char *fmt, ...) : 포맷문을 지정해서 새로운 문자열을 추가하는 API. /
파라미터 - Eina_Strbuf 객체, 포맷 형식, 3번째 부터는 포맷에 대입되는
데이터.

size_t eina_strbuf_length_get(Eina_Strbuf *buf) : Eina_Strbuf에 저장된
문자열의 길이를 반환하는 API.

Eina_Bool eina_strbuf_remove(Eina_Strbuf *buf, size_t start, size_t end) : 문자열의 일부분만을 삭제하는 API. / 파라미터 - Eina_Strbuf 객체,
삭제 범위 시작 위치, 삭제 범위 끝 위치.

void eina_strbuf_reset(Eina_Strbuf *buf) : Eina_Strbuf에 저장된 문자열을
초기화 하는 API.

Eina_Bool eina_strbuf_replace(Eina_Strbuf *buf, char *str, char *with,
unsigned int n) : 특정 문자를 다른 문자로 변경하는 함수API. /
파라미터 - Eina_Strbuf 객체, 변경할 문자, 변경될 문자, 변경 횟수.

int eina_strbuf_replace_all(Eina_Strbuf *buf, char *str, char *with) : 동일한
문자를 모두 치환하는 API함수. / 파라미터 - Eina_Strbuf 객체, 변경할
문자, 변경될 문자.

Eina_Bool eina_strbuf_insert(Eina_Strbuf *buf, char *str, size_t pos) : 문자열의 특정 위치에 문자를 삽입하는 API. / 파라미터 -
Eina_Strbuf 객체, 삽입할 문자열, 삽입 위치.

25. 배열 구조체 Eina_List

앱을 개발하다 보면 여러개의 문자열 혹은 사용자 데이터를 배열로 관리해야 하는 경우가 있습니다. EFL에서는 Eina_List라는 배열 구조체를 제공합니다. 예제를 통해서 사용방법을 알아보겠습니다.

1) List 위젯 생성

새로운 소스 프로젝트를 생성하고 Project name을 EinaListEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 새로운 코드를 추가합니다. appdata 구조체에 새로운 변수를 추가하고, 새로운 구조체를 정의하고, char* 배열에 10개의 문자열을 저장합니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    Evas_Object *list;  
    Evas_Object *button;  
    Eina_List *data_list;  
} appdata_s;
```

```
typedef struct {  
    appdata_s *ad;  
    char      *data;  
    int       id;  
} itemdata_s;
```

```
/* List */
const char *items[] = {
    "Seoul", "Tokyo", "New York", "London", "Beijing",
    "Moscow", "Singapore", "Busan", "Hong Kong", "Paris",
    NULL
};
```

list 는 List 위젯 변수입니다.

button은 삭제 Button 변수입니다.

data_list 는 배열 구조체 Eina_List의 변수입니다.

Itemdata_s는 List 위젯 이벤트 데이터용 구조체 입니다. ad에는 App 데이터를 저장하고, data에는 사용자가 선택한 문자열을 저장하고, id에는 선택된 항목 번호를 저장하겠습니다.

Items[]는 List 위젯에 추가하는 10개의 문자열을 저장합니다.

create_base_gui() 함수로 이동해서 새로운 코드를 추가합니다. Frame, Table, Button, List 위젯을 생성하는 코드입니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);
```

```

{

/* A frame surrounding the whole UI, for outer padding */
Evas_Object *frame;
frame = elm_frame_add(ad->conform);
evas_object_size_hint_weight_set(frame, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(frame, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(ad->conform, frame);
evas_object_show(frame);

/* A table to layout our objects */
Evas_Object *table;
table = elm_table_add(frame);
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(frame, table);
evas_object_show(table);
/* Set inner padding */
elm_table_padding_set(table, 5 * elm_scale_get(), 5 * elm_scale_get());


{
    /* Label */
    ad->label = elm_label_add(table);
    elm_object_text_set(ad->label, "Please select an item");
    evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND, 0);
    evas_object_size_hint_align_set(ad->label, EVAS_HINT_FILL, 0.5);
    elm_table_pack(table, ad->label, 0, 0, 1, 1);
    evas_object_show(ad->label);

    /* Minus button */
    ad->button = elm_button_add(ad->conform);
    elm_object_text_set(ad->button, "Remove");
    //evas_object_smart_callback_add(ad->button, "clicked", btn_clicked_cb,

```

```

ad);

    evas_object_size_hint_weight_set(ad->button, EVAS_HINT_EXPAND, 0);
    evas_object_size_hint_align_set(ad->button, EVAS_HINT_FILL, 0.5);
    elm_table_pack(table, ad->button, 1, 0, 1, 1);
    evas_object_show(ad->button);

    /* List view */
    ad->list = elm_list_add(ad->conform);
    elm_list_mode_set(ad->list, ELM_LIST_COMPRESS);
    evas_object_size_hint_weight_set(ad->list,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(ad->list,           EVAS_HINT_FILL,
EVAS_HINT_FILL);
    elm_table_pack(table, ad->list, 0, 1, 2, 1);
}

}

/* Let's add some elements to our lists */
populate_list(ad);

/* Go should be called before show for proper display */
elm_list_go(ad->list);
evas_object_show(ad->list);

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Frame에 Table을 추가하면 바깥쪽 여백을 지정할수 있습니다.

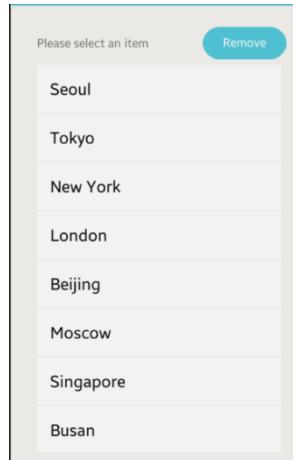
populate_list() 는 List 위젯에 10개의 텍스트 항목을 추가하는 함수입니다. 지금부터 만들어 보겠습니다. Create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```
static void
populate_list(appdata_s *ad)
{
    /* Now, let's create an Eina_List and add some data items to it */
    for (unsigned i = 0; items[i]; i++)
    {
        elm_list_item_append(ad->list, items[i], NULL, NULL, NULL, NULL);
    }

    elm_list_go(ad->list);
}
```

List에 10개의 텍스트 항목을 추가하는 코드입니다. 각 코드의 설명이 궁금하다면 ListEx 예제를 참조하기 바랍니다.

예제를 빌드하고 실행시켜 봅시다. 1개의 Label 위젯, Button 위젯, List 위젯이 생성되었고, List 위젯에는 10개의 텍스트 항목이 추가되었습니다.



2) Eina_List에 데이터 입력 & 출력

Eina_List에 문자열 데이터를 입력하고 출력하는 방법을 알아보겠습니다.
populate_list() 함수에 코드를 다음과 같이 수정합니다.

```
populate_list(appdata_s *ad)
{
    /* Now, let's create an Eina_List and add some data items to it */
    for (unsigned i = 0; items[i]; i++)
    {
        itemdata_s *idata = calloc(1, sizeof(itemdata_s));
        idata->ad = ad;
        idata->data = strdup(items[i]);
        idata->id = i + 1;
        ad->data_list = eina_list_append(ad->data_list, idata);
        //eli = elm_list_item_append(ad->list, items[i], NULL, NULL, NULL, NULL);

        itemdata_s *itemdata = eina_list_nth(ad->data_list, i);
        elm_list_item_append(ad->list, itemdata->data, NULL, NULL, NULL, NULL);
    }

    elm_list_go(ad->list);
}
```

Itemdata_s 구조체 변수에 모든 항목 데이터를 저장해서 Eina_List에 저장합니다. 그런 다음 Eina_List에 저장된 문자열을 List 위젯에 추가하는 코드입니다.

eina_list_append(Eina_List*, void*) 는 Eina_List에 새로운 항목을 추가하는 API입니다. Eina_List의 포인터를 반환하기 때문에 그 포인터 주소를 Eina_List 변수에 저장해야 합니다.

`eina_list_nth(Eina_List*, unsigned int)` 는 특정 위치에 있는 항목 데이터를 반환하는 API입니다.

`eina_list_count(Eina_List*)` 는 `Eina_List`에 저장된 항목 개수를 반환하는 API입니다.

예제를 다시 실행시켜 봅시다. 좀전과 동일한 목록이 나타납니다. 하지만 이제는 데이터가 전역변수에 저장되어 있기 때문에 어디서나 데이터를 사용할 수 있습니다.

3) 선택 항목 데이터를 화면에 표시

사용자가 `List` 위젯의 항목을 선택하면 항목 텍스트를 `Label` 위젯에 표시하는 기능을 구현해 보겠습니다. `populate_list()` 함수의 코드를 아래와 같이 수정합니다.

```
populate_list(appdata_s *ad)
{
    for (unsigned i = 0; items[i]; i++)
    {
        itemdata_s *idata = calloc(1, sizeof(itemdata_s));
        idata->ad = ad;
        idata->data = strdup(items[i]);
        idata->id = i + 1;
        ad->data_list = eina_list_append(ad->data_list, idata);
        //eli = elm_list_item_append(ad->list, items[i], NULL, NULL, NULL, NULL);

        itemdata_s *itemdata = eina_list_nth(ad->data_list, i);
        //elm_list_item_append(ad->list, itemdata->data, NULL, NULL, NULL, NULL);
        Elm_List_Item *eli;
```

```

    eli = elm_list_item_append(ad->list, itemdata->data, NULL, NULL,
list_item_clicked_cb, idata);
}

elm_list_go(ad->list);
}

```

List 위젯에 항목을 추가할 때 항목 선택 콜백 함수명을 지정하고 항목 데이터를 전달합니다.

항목 선택 콜백 함수명을 list_item_clicked 으로 지정했습니다.

콜백 함수를 정의할 차례입니다. populate_list() 함수 위에 새로운 함수를 생성합니다.

```

static void
list_item_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{
    itemdata_s *idata = data;
    char buf[256];

    snprintf(buf, 256, "%d. %s", idata->id, idata->data);
    elm_object_text_set(idata->ad->label, buf);
}

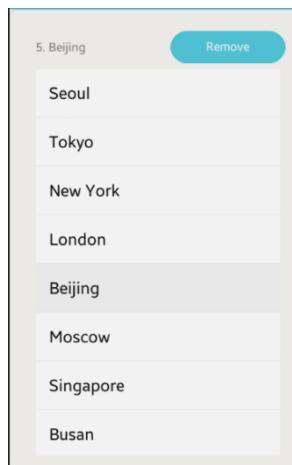
```

list_item_clicked_cb() 는 List 항목 선택 이벤트 함수입니다. 선택된 항목의 텍스트를 Label에 표시합니다.

sprintf() 함수를 사용해서 항목 번호와 텍스트를 하나의 문자열로 만듭니다.

`elm_object_text_set()` 함수를 사용해서 항목 데이터를 Label 위젯에 표시합니다.

예제를 다시 실행시켜 봅시다. List 위젯의 항목을 선택하면 항목 번호와 텍스트가 Label 위젯에 표시됩니다.



4) 데이터 삭제

Button을 클릭하면 현재 선택된 항목을 삭제하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 Button 생성 코드에 새로운 한줄을 추가합니다.

```
/* Minus button */
ad->button = elm_button_add(ad->conform);
elm_object_text_set(ad->button, "Remove");
evas_object_smart_callback_add(ad->button, "clicked", btn_clicked_cb, ad);
evas_object_size_hint_weight_set(ad->button, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(ad->button, EVAS_HINT_FILL, 0.5);
elm_table_pack(table, ad->button, 1, 0, 1, 1);
evas_object_show(ad->button);
```

Button의 콜백 함수명을 btn_clicked_cb 으로 지정했습니다.

이제 콜백 함수를 정의할 차례입니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    Elm_List_Item *it;

    if (!elm_list_items_get(ad->list))
    {
        elm_object_text_set(ad->button, "Remove");
        populate_list(ad);
        return;
    }

    it = elm_list_selected_item_get(ad->list);
    if (!it)
    {
        elm_object_text_set(ad->label, "No item selected");
        return;
    }

    /* Delete widget item, this will call item_del_cb */
    elm_object_item_del(it);

    /* If no more elements, offer to repopulate list */
    if (!ad->data_list)
    {
        elm_object_text_set(ad->button, "Populate");
        return;
    }
}
```

```
    }
}


```

List 위젯에 선택된 항목이 없다면 Button 위젯에 “Remove”라는 텍스트를 표시하고 함수를 빠져 나갑니다.

List 위젯에 선택된 항목이 존재한다면 해당 항목을 삭제하고 Button 위젯에 “Populate”라는 텍스트를 표시합니다.

List 위젯의 항목이 삭제될 때 배열 구조체에 저장된 항목 데이터도 삭제해야 합니다. `populate_list()` 함수 끝부분에 새로운 코드 1줄을 추가합니다.

```
itemdata_s *itemdata = eina_list_nth(ad->data_list, i);
Elm_List_Item *eli;
eli = elm_list_item_append(ad->list, itemdata->data, NULL, NULL,
list_item_clicked_cb, idata);
elm_object_item_del_cb_set(eli, item_del_cb);
}

elm_list_go(ad->list);
}
```

`elm_object_item_del_cb_set()` 는 항목 삭제 콜백 함수명을 지정하는 API입니다.

콜백 함수를 `item_del_cb`로 지정하였습니다. `elm_object_item_del_cb_set()` 함수 위에 콜백 함수를 생성합니다.

```

static void
item_del_cb(void *data, Evas_Object *obj, void *event_info)
{
    /* Those are the arguments */
    Elm_Widget_Item *it = event_info;
    itemdata_s *idata = data;
    (void) it;

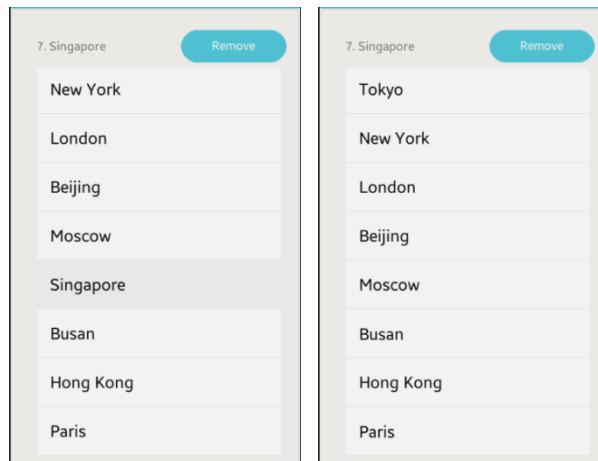
    /* Remove list element from Eina_List */
    idata->ad->data_list = eina_list_remove(idata->ad->data_list, idata);
    free(idata->data);
    free(idata);
}

```

`item_del_cb()` 는 List 항목 삭제 이벤트 함수입니다. `Eina_List`에 저장된 항목 데이터를 삭제합니다.

`eina_list_remove()` 는 `Eina_List`에 저장된 항목을 삭제하는 API입니다.

예제를 다시 실행시켜 봅시다. 항목을 하나 선택하고 Button을 클릭하면 해당 항목이 삭제됩니다.



5) 관련 API

Eina_List *eina_list_append(Eina_List *list, const void *data) : Eina_List에 새로운 항목을 추가하는 API. Eina_List의 포인터를 반환하기 때문에 그 포인터 주소를 Eina_List 변수에 저장해야 합니다. / 파라미터 - Eina_List 객체, 항목 데이터 객체.

unsigned int eina_list_count(const Eina_List *list) : Eina_List에 저장된 항목 개수를 반환하는 API. / 파라미터 - Eina_List 객체.

void *eina_list_nth(const Eina_List *list, unsigned int n) : 특정 위치에 있는 항목 데이터를 반환하는 API. / 파라미터 - Eina_List 객체, 항목 인덱스 번호.

Eina_List *eina_list_remove(Eina_List *list, const void *data) : Eina_List에 저장된 항목을 삭제하는 API. Eina_List의 포인터를 반환하기 때문에 그 포인터 주소를 Eina_List 변수에 저장해야 합니다. / 파라미터 - Eina_List 객체, 항목 데이터 객체.

void elm_list_clear(Evas_Object *obj) : List 위젯의 목록을 모두 삭제하는 API.

26. 타이머 사용방법

일정한 시간 주기로 이벤트가 발생하는 타이머는 알람앱 및 애니메이션 효과에 필수적으로 사용됩니다. 게임 앱을 개발할 때는 동시에 여러개의 타이머가 구동됩니다. 예제를 통해서 사용방법을 알아보겠습니다.

1) 타이머 이벤트 시작

새로운 소스 프로젝트를 생성하고 Project name을 TimerEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 appdata 구조체에 새로운 변수를 추가합니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    Ecore_Timer *timer1;  
    int timer_count;  
} appdata_s;
```

Ecore_Timer 는 타이머 구조체 입니다.

timer_count 는 타이머 이벤트 발생 회수를 저장하는 변수입니다.

create_base_gui() 함수 위에 2개의 함수를 생성합니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child, double h_weight, double v_weight,
```

```

        double h_align, double v_align)
{
    /* we use a frame for padding only */
    Evas_Object *frame = elm_frame_add(box);
    elm_object_style_set(frame, "pad_small");
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    elm_box_pack_end(box, frame);
    evas_object_show(frame);
}

static Evas_Object *
my_button_add(Evas_Object *parent, const char *text, Evas_Smart_Cb cb, void *cb_data)
{
    Evas_Object *btn;

    btn = elm_button_add(parent);
    elm_object_text_set(btn, text);
    evas_object_smart_callback_add(btn, "clicked", cb, cb_data);

    return btn;
}

```

my_box_pack() 은 Box에 위젯을 추가하는 함수입니다.

my_button_add() 은 Button 위젯을 생성해서 반환하는 함수입니다.

`create_base_gui()` 함수로 이동해서 새로운 코드를 추가합니다. Box와 2개의 Button을 생성하는 코드입니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

Evas_Object *btn, *box;

/* Container: standard box */
box = elm_box_add(ad->win);
elm_box_horizontal_set(box, EINA_FALSE);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(box, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* Label */
ad->label = elm_label_add(box);
elm_object_text_set(ad->label, "No timer");
my_box_pack(box, ad->label, EVAS_HINT_EXPAND, 0.0, 0.5, 0.0);

/* Button-1 */
btn = my_button_add(box, "Start", btn_start_cb, ad);
my_box_pack(box, btn, EVAS_HINT_EXPAND, 0, EVAS_HINT_FILL,
```

```

EVAS_HINT_FILL);

/* Button-2 */
btn = my_button_add(box, "Stop", btn_stop_cb, ad);
my_box_pack(box, btn, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND,
EVAS_HINT_FILL, 0.0);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

1번째 Button은 타이머를 시작하는 기능을 담당합니다. Button 콜백 함수를 만들겠습니다. create_base_gui() 함수 위에 2개의 함수를 생성합니다.

```

static void
btn_start_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    ad->timer_count = 0;
    ad->timer1 = ecore_timer_add(1.0, timer1_cb, ad);
    elm_object_text_set(ad->label, "Timer started");
}

```

```

static void
btn_stop_cb(void *data, Evas_Object *obj, void *event_info)
{
}

```

1번째 Button을 클릭하면 btn_start_cb() 함수가 호출됩니다. 타이머 발생

회수를 0으로 초기화하고, 새로운 타이머를 생성 & 시작하는 코드입니다.

`ecore_timer_add(double, Ecore_Task_Cb, void*)` 는 새로운 타이머를 생성해서 반환하는 API입니다. 1번째 파라미터는 시간 간격입니다. 단위는 Second 입니다. 만약 1.5초 마다 이벤트가 발생하려면 1.5를 전달하면 됩니다. 2번째 파라미터는 타이머 이벤트 콜백 함수명, 3번째는 사용자 데이터입니다.

`btn_stop_cb()` 는 2번째 Button의 이벤트 함수입니다. 잠시 후에 기능을 추가해 보겠습니다.

이제 타이머 이벤트 콜백 함수를 생성할 차례입니다. `btn_start_cb()` 함수 위에 새로운 함수를 추가합니다.

```
static Eina_Bool  
timer1_cb(void *data EINA_UNUSED)  
{  
    appdata_s *ad = data;  
    ad->timer_count ++;  
    char buf[100];  
    sprintf(buf, "Count - %d", ad->timer_count);  
  
    elm_object_text_set(ad->label, buf);  
    return ECORE_CALLBACK_RENEW;  
}
```

타이머 이벤트가 발생하면 위 함수가 호출됩니다. 타이머 회수를 증가하고, 현재 카운트를 화면에 표시하는 코드입니다.

예제를 빌드하고 실행시켜 봅시다. Start 버튼을 누르면 1초에 한번씩

Label 위젯에 숫자가 증가합니다.



2) 타이머 중지

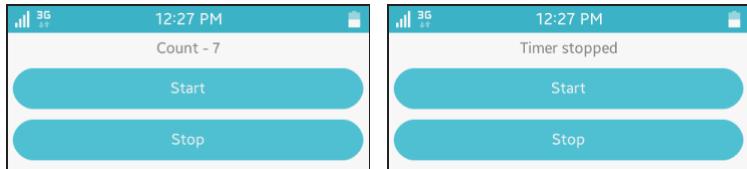
Stop 버튼을 누르면 타이머가 중지되는 기능을 구현해 보겠습니다.
btn_stop_cb() 함수에 아래와 같이 코드를 추가합니다.

```
static void  
btn_stop_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    ecore_timer_freeze(ad->timer1);  
    ecore_timer_del(ad->timer1);  
    elm_object_text_set(ad->label, "Timer stopped");  
}
```

ecore_timer_freeze(Ecore_Timer*) 는 타이머 이벤트를 일시정지하는 API입니다. 다시 재시작하려면 ecore_timer_thaw() 함수를 사용하면 됩니다.

ecore_timer_del(Ecore_Timer*) 는 타이머 객체를 삭제하는 API입니다.

예제를 다시 실행합시다. Start 버튼을 눌렀다가 잠시 후에 Stop 버튼을 누르면 타이머 이벤트가 중지됩니다.



3) 관련 API

`Ecore_Timer *ecore_timer_add(double in, Ecore_Task_Cb func, void *data)` : 새로운 타이머를 생성해서 반환하는 API. / 파라미터 - 시간 간격(단위는 Second), 타이머 이벤트 콜백 함수명, 사용자 데이터.

`void ecore_timer_freeze(Ecore_Timer *timer)` : 타이머 이벤트 일시정지 API.

`void ecore_timer_thaw(Ecore_Timer *timer)` : 타이머 재가동 API.

`void *ecore_timer_del(Ecore_Timer *timer)` : 타이머 객체를 삭제하는 API.

27. 날짜 & 시간

i18n_ucalendar_h를 사용하면 현재 날짜와 시간을 구할 수 있으며, 2개의 시간을 더하는 계산도 할 수 있습니다. 예제를 통해서 사용방법을 알아보겠습니다.

1) TimeZone 구하기

새로운 소스 프로젝트를 생성하고 Project name을 DateTime 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 appdata 구조체에 새로운 변수를 추가합니다. 라이브러리 헤더파일도 인클루드 합니다.

```
#include "datetime.h"
#include <utils_i18n.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label1;
    Evas_Object *label2;
    Evas_Object *label3;
    Evas_Object *label4;
    Evas_Object *label5;
    Evas_Object *slide;
    Ecore_Timer *timer;
    char *tzid;
    i18n_ucalendar_h ucal;
} appdata_s;
```

label1에는 TimeZone을 표시하고, label2에는 현재 날짜와 시간을 표시하고, label3에는 Posix 시간을 표시하고, label4에는 시간 더하기 제목, label5에는 시간 더하기 결과를 표시하겠습니다.
slide는 덧셈 날짜를 변경하는 슬라이드 위젯입니다.

tzid 는 TimeZone을 저장하는 문자열 변수입니다.

i18n_ucalendar_h 는 시간&날짜 정보를 저장하는 구조체 입니다.

화면에 8개의 Label 위젯을 생성하겠습니다. 현재 시간을 구하려면 TimeZone 설정도 구해야 합니다. create_base_gui() 함수로 이동해서 새로운 코드를 추가합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Table */
Evas_Object *table = elm_table_add(ad->win);
elm_table_padding_set(table, 5 * elm_scale_get(), 5 * elm_scale_get());
elm_object_content_set(ad->conform, table);
evas_object_show(table);

{
    Evas_Object *o;

    /* Timezone label */
```

```

o = elm_label_add(table);
elm_object_text_set(o, "Time Zone:");
table_pack(table, o, 0, 0, 1, 1, 0.5, 1.0, 1.0, 1.0);

ad->label1 = elm_label_add(table);
table_pack(table, ad->label1, 1, 0, 1, 1, 0.5, 1.0, 0.0, 1.0);

system_settings_get_value_string(SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE, &ad-
>tzid);
elm_object_text_set(ad->label1, ad->tzid);

/* Current time label */
o = elm_label_add(table);
elm_object_text_set(o, "Current Time:");
table_pack(table, o, 0, 1, 1, 1, 0.5, 0.0, 1.0, 0.5);

ad->label2 = elm_label_add(table);
table_pack(table, ad->label2, 1, 1, 1, 1, 0.5, 0.0, 0.0, 0.5);

/* Current time label */
o = elm_label_add(table);
elm_object_text_set(o, "Since Epoch:");
table_pack(table, o, 0, 2, 1, 1, 0.5, 0.0, 1.0, 0.5);

ad->label3 = elm_label_add(table);
table_pack(table, ad->label3, 1, 2, 1, 1, 0.5, 0.0, 0.0, 0.5);

/* Showcase datetime computation */
ad->label4 = elm_label_add(table);
elm_object_text_set(ad->label4, "40 days later:");
table_pack(table, ad->label4, 0, 3, 1, 1, 0.5, 0.0, 1.0, 0.5);

ad->label5 = elm_label_add(table);
table_pack(table, ad->label5, 1, 3, 1, 1, 0.5, 0.0, 0.0, 0.5);

```

```
}

/* Show window after base gui is set up */
evas_object_show(ad->win);
```

4개의 Label 과 2개의 Button을 생성하였습니다.

system_settings_get_value_string() 시스템 설정 정보를 구하는 API입니다. 1번째 파라미터에 SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE를 전달하면 2번째 파라미터에 TimeZone 문자열이 반환됩니다.

예제를 빌드하고 실행시켜 봅시다. 환경설정에 지정된 TimeZone 이 Label1에 표시됩니다.



2) 현재 시간 생성

i18n_ucalendar_h 객체를 생성하면 자동으로 현재 시간이 저장됩니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static i18n_ucalendar_h
create_time(char *tzid)
{
    i18n_ucalendar_h ucal;
    i18n_uchar *_tzid = (i18n_uchar*)calloc(strlen(tzid) + 1, sizeof(i18n_uchar));
```

```

// converts 'tzid' to unicode string
i18n_ustring_copy_ua(_tzid, tzid);
// gets length of '_tzid'
int len = i18n_ustring_get_length(_tzid);
// creates i18n_ucalendar_h
int ret = i18n_ucalendar_create(_tzid, len, "en_US", I18N_UCALENDAR_TRADITIONAL,
&ucal);
if (ret != 0)
{
    dlog_print(DLOG_ERROR, LOG_TAG, "i18n_ucalendar_create() failed with err = %d", ret);
    return NULL;
}

return ucal;
}

```

i18n_ucalendar_h 객체를 생성하려면 TimeZone을 i18n_uchar 형식으로 변경해야 합니다.

i18n_ustring_copy_ua() 는 char 배열에 저장된 TimeZone을 i18n_uchar 배열에 복사하는 API입니다.

i18n_ustring_get_length() 는 i18n_uchar 배열의 길이를 반환하는 API입니다.

i18n_ucalendar_create(i18n_uchar *zone_id, int32_t len, char *locale, i18n_ucalendar_type_e type, i18n_ucalendar_h *calendar) 는 i18n_ucalendar_h 객체를 생성하는 API입니다. 1번째 파라미터는 TimeZone, 2번째는 TimeZone 문자열 길이, 3번째는 지역명, 4번째는 ucalendar 타입, 5번째는 i18n_ucalendar_h 객체가 반환됩니다.

이렇게 해서 현재 시간을 구할 수 있습니다. i18n_ucalendar_h에 저장된 날짜와 시간을 문자열로 변경해서 화면에 출력해 보겠습니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다. i18n_ucalendar_h 객체를 받아서 날짜와 시간을 문자열로 변경하는 함수입니다.

```
static void
update(appdata_s *ad)
{
    int year, month, day, hour, minute, second;
    i18n_update udate;
    char buf[256];
    int diff;

    /* Current time */
    i18n_ucalendar_get_now(&udate);
    i18n_ucalendar_set_milliseconds(ad->ucal, udate);
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_YEAR, &year);
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_MONTH, &month);
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_DATE, &day);
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_HOUR_OF_DAY, &hour);
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_MINUTE, &minute);
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_SECOND, &second);
    snprintf(buf, sizeof(buf), "%d/%02d/%02d %02d:%02d:%02d", year, month + 1, day,
hour, minute, second);
    elm_object_text_set(ad->label2, buf);
}
```

i18n_ucalendar_get(i18n_ucalendar_h, i18n_ucalendar_date_fields_e, int32_t) 는 i18n_ucalendar_h에서 한가지 데이터를 구하는 API입니다.

2번째 파라미터에 I18N_UCALENDAR_YEAR를 전달하면 3번째
파라미터에서 연도값이 반환됩니다.

순서대로 년, 월, 일, 시, 분, 초 값을 구합니다.

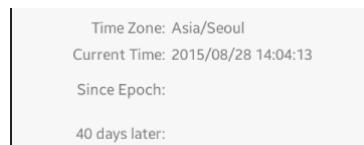
sprintf()를 사용해서 6가지 시간 값을 하나의 문자열로 만들어서
반환합니다. 주의해야 할 것이 있는데 month 는 1을 더해주어야 한다는
것입니다. month의 범위는 0~11 사이입니다.

방금 만든 2개의 함수를 사용해서 현재 시간을 화면에 표시해
보겠습니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

/* Prepare calendar object and 1-second timer */
ad->ucal = create_time(ad->tzid);
update(ad);
}
```

예제를 다시 실행시켜 봅시다. 현재 날짜와 시간이 Label2에 표시됩니다.



3) 디지털 시계

Label2에 표시된 날짜와 시간이 1초에 한번씩 갱신되는 기능을 구현해 보겠습니다. 타이머를 이용하여 디지털 시계와 동일한 기능을 구현하는 것입니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

/* Prepare calendar object and 1-second timer */
ad->ucal = create_time(ad->tzid);
ad->timer = ecore_timer_add(1.0, timer_cb, ad);
update(ad);
}
```

1초에 한번씩 타이머를 발생시키는 코드입니다. 타이머 이벤트 함수를 만들 차례입니다. `create_base_gui()` 함수 위에 새로운 함수를 생성합니다.

```
static Eina_Bool
timer_cb(void *data)
{
    appdata_s *ad = data;
    update(ad);
    return ECORE_CALLBACK_RENEW;
}
```

타이머 이벤트가 발생하면 `update()` 함수를 호출해서 날짜&시간을 갱신하는 코드입니다. 예제를 다시 실행시켜 봅시다. 1초마다 시간이 변경됩니다.

Time Zone: Asia/Seoul Current Time: 2015/08/28 14:15:35 Since Epoch: 40 days later:	Time Zone: Asia/Seoul Current Time: 2015/08/28 14:16:04 Since Epoch: 40 days later:
--	--

4) posix 시간 구하기

컴퓨터는 밀리세컨 단위를 사용하는데 기원 1년부터 현재까지의 시간을 밀리세컨 단위로 환산하면 엄청난 숫자가 만들어 집니다. 그래서 컴퓨터가 출현한 이후를 기준으로 새로운 시간 체계가 필요해 졌습니다. 그래서 정한 날짜가 1970년 1월 1일 0시입니다. 이것을 posix 시간이라고 합니다. posix 시간을 이용하면 앱이 실행된 이후에 얼마나 시간이 지났는지를 계산할 수 있습니다.

update() 함수에 새로운 코드 한줄을 추가합니다. Button 콜백 함수명을 지정하는 코드입니다.

```

static void
update(appdata_s *ad)
{
    int year, month, day, hour, minute, second;
    i18n_update udate;
    char buf[256];
    int diff;

    /* Current time */
    i18n_ucalendar_get_now(&udate);
    i18n_ucalendar_set_milliseconds(ad->ucal, udate);
    ~
    elm_object_text_set(ad->label2, buf);

    /* POSIX time since EPOCH */
    sprintf(buf, sizeof(buf), "%llums", (unsigned long long) udate);

```

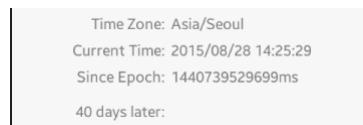
```
    elm_object_text_set(ad->label3, buf);  
}  
_____
```

i18n_update 는 double 과 동일한 데이터 타입입니다. posix 시간은 매우 큰 숫자이기 때문에 이런 형식을 사용합니다.

i18n_ucalendar_get_now(i18n_update) 는 현재 시간을 밀리세컨 단위로 반환하는 API 입니다.

i18n_ucalendar_set_milliseconds(i18n_ucalendar_h, i18n_update) 는 i18n_ucalendar_h에 새로운 시간을 지정을 하는 API 입니다.

예제를 다시 실행면 label3에 Posix 시간이 표시됩니다. 단위는 밀리세컨입니다.



4) 시간 계산

날짜를 더하기 계산하는 방법을 알아보겠습니다. 슬라이더 위젯을 하나 추가하겠습니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
ad->label5 = elm_label_add(table);  
table_pack(table, ad->label5, 1, 3, 1, 1, 0.5, 0.0, 0.0, 0.5);
```

```
/* Spinner for more time difference */  
ad->slide = elm_slider_add(table);
```

```

elm_slider_min_max_set(ad->slide, -365, 365);
elm_slider_value_set(ad->slide, 40);
table_pack(table, ad->slide, 0, 4, 2, 1, 1.0, 2.0, -1.0, 0.0);
evas_object_smart_callback_add(ad->slide, "changed", spinner_cb, ad);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Slider 위젯을 생성하고 범위를 -365 부터 +365 까지 지정하였습니다.
Value 는 40을 지정하였습니다.

Slider의 이벤트 함수를 만들 차례입니다. `create_base_gui()` 함수 위에 새로운 함수를 생성합니다. Slider의 Value가 변경되면 `update()` 함수가 호출됩니다.

```

static void
spinner_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    update(ad);
}

```

마지막으로 `update()` 함수 끝부분에 새로운 코드를 추가합니다.

```

static void
update(appdata_s *ad)
{
    ~

```

```

/* POSIX time since EPOCH */
snprintf(buf, sizeof(buf), "%llums", (unsigned long long) update);
elm_object_text_set(ad->label3, buf);

/* 40 days later (label) */
diff = (int) elm_slider_value_get(ad->slide);
if (diff >= 0)
    snprintf(buf, sizeof(buf), "%d day%s later:", diff, (diff > 1) ? "s" : "");
else if (diff < 0)
    snprintf(buf, sizeof(buf), "%d day%s earlier:", -diff, (diff < -1) ? "s" : "");
elm_object_text_set(ad->label4, buf);

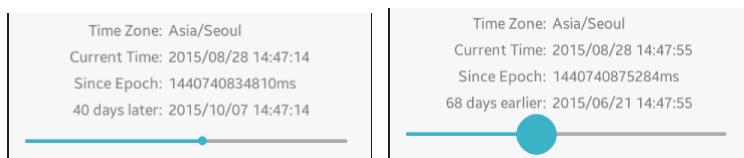
/* 40 days later (value) */
i18n_ucalendar_add(ad->ucal, I18N_UCALENDAR_DATE, diff);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_YEAR, &year);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_MONTH, &month);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_DATE, &day);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_HOUR_OF_DAY, &hour);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_MINUTE, &minute);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_SECOND, &second);
snprintf(buf, sizeof(buf), "%d/%02d/%02d %02d:%02d:%02d", year, month + 1,
day, hour, minute, second);
elm_object_text_set(ad->label5, buf);
}

```

Slider의 Value 값을 양수와 음수로 구분해서 label4에 출력하고, 현재 날짜와 덧셈한 결과를 label5에 출력하는 코드입니다.

i18n_ucalendar_add() 는 i18n_ucalendar_h 객체의 특정 항목에 숫자를 더하는 함수입니다. 2번째 파라미터에 I18N_UCALENDAR_DATE를 전달하고, 3번째 파라미터에 40을 전달하면 40일 이후 날짜가 만들어 집니다.

예제를 다시 실행시켜서 2번째 Button을 클릭해 봅시다. 오늘로부터 40일 후의 날짜가 Label4에 표시됩니다. Slider를 드래그하면 계산에 사용되는 날짜가 변경됩니다.



5) 관련 API

`int system_settings_get_value_string(system_settings_key_e key, char **value)` : 시스템 설정 정보를 구하는 API. 1번째 파라미터에 `SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE`를 전달하면 2번째 파라미터에 `TimeZone` 문자열이 반환됩니다.

`i18n_uchar* i18n_ustring_copy_ua (i18n_uchar *dest, const char *src)` : `char` 배열에 저장된 `TimeZone`을 `i18n_uchar` 배열에 복사하는 API.

`int32_t i18n_ustring_get_length (i18n_uchar *s)` : `i18n_uchar` 배열의 길이를 반환하는 API.

`int i18n_ucalendar_create (i18n_uchar *zone_id, int32_t len, char *locale, i18n_ucalendar_type_e type, i18n_ucalendar_h *calendar)` : `i18n_ucalendar_h` 객체를 생성하는 API. / 파라미터 - `TimeZone`, `TimeZone` 문자열 길이, 지역명, ucalendar 타입, `i18n_ucalendar_h` 객체를 반환.

```
int i18n_ucalendar_get( i18n_ucalendar_h calendar,
i18n_ucalendar_date_fields_e field, int32_t *val ) : i18n_ucalendar_h에서
한가지 데이터를 구하는 API. /
파라미터 - i18n_ucalendar_h 객체, 날짜&시간 필드, 날짜&시간 값 반환.
```

날짜&시간 필드 종류

- I18N_UCALENDAR_YEAR : 연도
- I18N_UCALENDAR_MONTH : 월
- I18N_UCALENDAR_DATE : 일
- I18N_UCALENDAR_DAY_OF_WEEK : 요일
- I18N_UCALENDAR_AM_PM : 오전 or 오후
- I18N_UCALENDAR_HOUR : 시
- I18N_UCALENDAR_MINUTE : 분
- I18N_UCALENDAR_SECOND : 초
- I18N_UCALENDAR_MILLISECOND : 밀리세컨

i18n_update : double 과 동일한 데이터 타입. posix 시간을 저장할때 사용합니다.

```
int i18n_ucalendar_get_milliseconds( i18n_ucalendar_h calendar,
i18n_update *date ) : i18n_ucalendar_h에 저장된 시간을 posix 시간으로
변경하는 API. 단위는 밀리세컨.
```

```
int i18n_ucalendar_add( i18n_ucalendar_h calendar,
i18n_ucalendar_date_fields_e field, int32_t amount ) : i18n_ucalendar_h
객체의 특정 항목에 숫자를 더하는 API. /
파라미터
- i18n_ucalendar_h 객체, 날짜&시간 필드, 더해지는 숫자.
```

28. Calendar 예제

이번 시간에는 `i18n_ucalendar_h`를 사용해서 달력을 만드는 방법을 알아보겠습니다.

1) 화면 UI 구성

새로운 소스 프로젝트를 생성하고 Project name을 `CalendarEx` 으로 지정합니다. 소스 프로젝트가 생성되었으면 `src` 폴더에 소스파일(~.c)을 열고 `appdata` 구조체에 새로운 변수를 추가하고 라이브러리 헤더파일도 추가합니다.

```
#include "calendarex.h"
#include <utils_i18n.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    Evas_Object *label_day[6][7];
    char *tzid;
    i18n_ucalendar_h ucal;
} appdata_s;
```

`label_day` 은 날짜를 표시하는 Label 위젯의 배열입니다. 수평으로 7칸, 수직으로 6줄까지 날짜를 표시할 수 있습니다.

tzid에는 TimeZone을 저장합니다.

ucal에는 오늘 날짜와 시간을 저장합니다.

화면에 날짜 표시용 Label 위젯들을 추가하겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다. Table에 위젯을 추가하는 함수입니다.

```
static void  
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)  
{  
    evas_object_size_hint_align_set(child, 0.5, 0.5);  
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);  
    elm_table_pack(table, child, x, y, w, h);  
    evas_object_show(child);  
}
```

그런 다음 `create_base_gui()` 끝부분에 새로운 코드를 추가합니다. Box와 Table을 생성하고 Label을 추가하는 코드입니다. Conformant 생성 코드는 주석 처리합니다.

```
/* Conformant */  
/*ad->conform = elm_conformant_add(ad->win);  
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);  
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);  
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
elm_win_resize_object_add(ad->win, ad->conform);  
evas_object_show(ad->conform);*/  
  
/* Box to put the table in so we can bottom-align the table
```

```

        /* window will stretch all resize object content to win size */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, box);
evas_object_show(box);

/* Table */
Evas_Object *table = elm_table_add(ad->win);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 10 * elm_config_scale_get(), 10 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fiill width but align to bottom of box */
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, 0.0);
elm_box_pack_end(box, table);
evas_object_show(table);

{
    /* Label*/
ad->label = elm_label_add(ad->win);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
my_table_pack(table, ad->label, 0, 0, 7, 1);

for(int j=0; j < 6; j++)
{
    for(int i=0; i < 7; i++)
    {
        ad->label_day[j][i] = elm_label_add(ad->win);
        elm_object_text_set(ad->label_day[j][i], ".");
        my_table_pack(table, ad->label_day[j][i], i, j + 2, 1, 1);
    }
}

```

```
        }
    }

/* Show window after base gui is set up */
evas_object_show(ad->win);
```

2개의 for 루프를 사용해서 수평으로 7칸, 수직으로 6칸의 Label 위젯을 생성하였습니다.

예제를 빌드하고 실행시켜 봅시다. 새로 추가된 Label에 '.' 기호가 표시되었습니다.



2) 오늘 날짜&시간 표시

i18n_ucalendar_h 객체를 생성해서 오늘 날짜와 시간을 화면에 표시해 보겠습니다. create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

```
static i18n_ucalendar_h
create_time(char *tzid)
{
    i18n_ucalendar_h ucal;
    i18n_uchar *_tzid = (i18n_uchar*)calloc(strlen(tzid) + 1, sizeof(i18n_uchar));
```

```

i18n_ustring_copy_ua(_tzid, _tzid);
int len = i18n_ustring_get_length(_tzid);
int ret = i18n_ucalendar_create(_tzid, len, "en_US", I18N_UCALENDAR_TRADITIONAL,
&ucal);
return ucal;
}

static char* time2string(i18n_ucalendar_h ucal)
{
    int year, month, day, hour, minute, second;
    i18n_ucalendar_get(ucal, I18N_UCALENDAR_YEAR, &year);
    i18n_ucalendar_get(ucal, I18N_UCALENDAR_MONTH, &month);
    i18n_ucalendar_get(ucal, I18N_UCALENDAR_DATE, &day);
    i18n_ucalendar_get(ucal, I18N_UCALENDAR_HOUR, &hour);
    i18n_ucalendar_get(ucal, I18N_UCALENDAR_MINUTE, &minute);
    i18n_ucalendar_get(ucal, I18N_UCALENDAR_SECOND, &second);

    char *buf = malloc(100);
    sprintf(buf, "Now :%04d-%02d-%02d %02d:%02d:%02d", year, month + 1, day, hour,
minute, second);
    return buf;
}

```

create_time() 는 i18n_ucalendar_h 객체를 생성해서 반환하는 함수이고, time2string() 는 i18n_ucalendar_h에 저장된 날짜를 문자열로 변경해서 반환하는 함수입니다. 자세한 설명은 DateTime 예제를 참조하기 바랍니다.

이제 위 함수를 사용해서 오늘 날짜와 시간을 화면에 표시해 보겠습니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
evas_object_show(ad->win);
```

```
    system_settings_get_value_string(SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE,
&ad->tzid);
    ad->ucal = create_time(ad->tzid);
    char *buf = time2string(ad->ucal);
    elm_object_text_set(ad->label, buf);
}
```

system_settings_get_value_string() 함수를 사용해서 TimeZone을 구합니다.

create_time() 함수를 사용해서 i18n_ucalendar_h 객체를 생성합니다.

time2string() 함수를 사용해서 i18n_ucalendar_h에 저장된 날짜와 시간을 하나의 문자열로 변경합니다.

예제를 다시 실행시켜 봅시다. 현재 날짜와 시간이 Label 위젯에 표시되었습니다.

Now : 2015-08-28 03:12:08

3) 달력 날짜 계산

이번달 1일이 무슨 요일인지를 파악하고 Label 배열에 날짜를 입력해 보겠습니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void
draw_calendar(appdata_s *ad)
{
```

```

int date, month, dow, days, is_leap;
int max_day[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

i18n_ucalendar_set(ad->ucal, I18N_UCALENDAR_DATE, 1);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_MONTH, &month);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_DAY_OF_WEEK, &dow);
days = max_day[month];

if( month == 1 )
{
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_IS_LEAP_MONTH, &is_leap);
    if( is_leap == 1 )
        days = 29;
}

int i=0, j=0;
char buf[10];

i = dow - 1;
for(int d=1; d <= days; d++)
{
    sprintf(buf, "%d", d);
    elm_object_text_set(ad->label_day[j][i], buf);

    i++;
    if( i >= 7 )
    {
        i = 0;
        j++;
    }
}

```

max_day[] 는 1월부터 12월까지 최대 날짜를 저장해두는 배열입니다.

i18n_ucalendar_set() 는 i18n_ucalendar_h 객체의 특정 항목에 새로운 값을 지정하는 API입니다. 2번째 파라미터에 I18N_UCALENDAR_DATE을 전달하고 3번째 파라미터에 1을 전달하면, 날짜를 1일로 변경하게 됩니다.

i18n_ucalendar_get() 는 i18n_ucalendar_h 객체의 특정 항목 값을 반환하는 API입니다. 날짜 항목의 종류는 다음과 같습니다.

- 2번째 파라미터에 I18N_UCALENDAR_MONTH를 전달하면 3번째 파라미터에서 Month 값을 반환합니다.
- 2번째 파라미터에 I18N_UCALENDAR_DAY_OF_WEEK를 전달하면 3번째 파라미터에서 요일 번호를 반환합니다.
- 2번째 파라미터에 I18N_UCALENDAR_IS_LEAP_MONTH를 전달하면 3번째 파라미터에서 윤달인지 여부를 반환합니다.

만약 이번달이 윤달이라면 최대 날짜를 29일로 변경합니다.

그 이후는 1일부터 마지막 날짜 까지 숫자를 문자열로 변경해서 해당되는 Label 위젯에 표시하는 코드입니다.

위 함수를 앱이 실행될 때 호출해 주면 됩니다. create_base_gui() 함수 끝에 새로운 코드 한줄을 추가합니다.

```
elm_object_text_set(ad->label, buf);

draw_calendar(ad);

}
```

예제를 다시 실행시켜 봅시다. 오늘 날짜에 해당하는 달력이 표시되었습니다.

Now : 2015-08-28 03:22:58							
.	1
2	3	4	5	6	7	8	
9	10	11	12	13	14	15	
16	17	18	19	20	21	22	
23	24	25	26	27	28	29	
30	31	

4) 오늘 날짜 표시

오늘 날짜에 해당하는 숫자를 '[']' 기호로 묶는 기능을 구현해 보겠습니다. draw_calendar() 함수에 새로운 코드를 추가합니다.

```

static void
draw_calendar(appdata_s *ad)
{
    int date, month, dow, days, is_leap;
    int max_day[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_DATE, &date);
i18n_ucalendar_set(ad->ucal, I18N_UCALENDAR_DATE, 1);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_MONTH, &month);
i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_DAY_OF_WEEK, &dow);
days = max_day[month];

if( month == 1 )
{
    i18n_ucalendar_get(ad->ucal, I18N_UCALENDAR_IS_LEAP_MONTH, &is_leap);
    if( is_leap == 1 )
        days = 29;
}

int i=0, j=0;
```

```

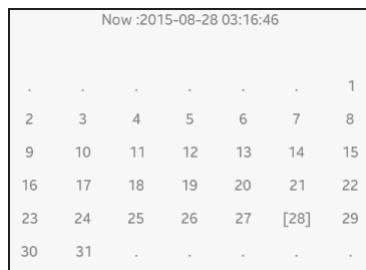
char buf[10];

i = dow - 1;
for(int d=1; d <= days; d++)
{
    sprintf(buf, "%d", d);
if( d == date )
    sprintf(buf, "[%d]", d);
elm_object_text_set(ad->label_day[j][i], buf);
~
```

i18n_ucalendar_get() 함수를 사용해서 오늘 날짜를 구해서 변수에 저장합니다.

for 루프 안에서 오늘 날짜와 동일한 순서가 돌아오면 '['' 기호를 추가합니다.

예제를 다시 실행시켜 봅시다. 오늘 날짜와 다른 날짜가 구분 되었습니다.



5) Calendar 위젯

Calendar 위젯을 사용하면 간편하게 Calendar를 구현할 수 있습니다. 새로운 예제를 생성하고 이름을 CalendarWidgetEx 이라고 지정합니다.

create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
/* Label*/
ad->label = elm_label_add(ad->win);
elm_object_text_set(ad->label, "<align=center>Calendar</>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND, 0.0);
evas_object_size_hint_align_set(ad->label, EVAS_HINT_FILL, 0.5);
elm_box_pack_end(box, ad->label);
evas_object_show(ad->label);

Evas_Object *cal = elm_calendar_add(ad->win);
evas_object_size_hint_weight_set(cal, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(cal, EVAS_HINT_FILL, 0.5);
elm_box_pack_end(box, cal);
evas_object_show(cal);
```

```

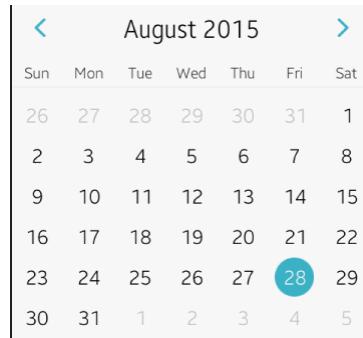
        }
    }

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

elm_calendar_add() 는 Caledar 위젯을 생성하는 API 입니다.

예제를 실행하면 Caledar가 출력됩니다. 좌우 화살표를 누르면 이전달과 다음달로 이동할수 있습니다.



Calendar 위젯의 자세한 사용방법은 Help Contents에서 다음 경로를 참조하기 바랍니다.

API References > Native Application > Mobile Native > Native API Reference > UI > EFL > Elementary > Elementary Widgets

6) 관련 API

int system_settings_get_value_string(system_settings_key_e key, char **value) : 시스템 설정 정보를 구하는 API. 1번째 파라미터에 SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE를 전달하면 2번째

파라미터에 TimeZone 문자열이 반환됩니다.

`i18n_uchar* i18n_ustring_copy_ua (i18n_uchar *dest, const char *src)` : char 배열에 저장된 TimeZone을 i18n_uchar 배열에 복사하는 API.

`int32_t i18n_ustring_get_length (i18n_uchar *s)` : i18n_uchar 배열의 길이를 반환하는 API.

`int i18n_ucalendar_create (i18n_uchar *zone_id, int32_t len, char *locale, i18n_ucalendar_type_e type, i18n_ucalendar_h *calendar)` : i18n_ucalendar_h 객체를 생성하는 API. / 파라미터 - TimeZone, TimeZone 문자열 길이, 지역명, ucalendar 타입, i18n_ucalendar_h 객체를 반환.

`int i18n_ucalendar_get (i18n_ucalendar_h calendar, i18n_ucalendar_date_fields_e field, int32_t *val)` : i18n_ucalendar_h에서 한가지 데이터를 구하는 API. / 파라미터 - i18n_ucalendar_h 객체, 날짜&시간 필드, 날짜&시간 값 반환.

날짜&시간 필드 종류

- I18N_UCALENDAR_YEAR : 연도
- I18N_UCALENDAR_MONTH : 월
- I18N_UCALENDAR_DATE : 일
- I18N_UCALENDAR_DAY_OF_WEEK : 요일
- I18N_UCALENDAR_AM_PM : 오전 or 오후
- I18N_UCALENDAR_HOUR : 시
- I18N_UCALENDAR_MINUTE : 분
- I18N_UCALENDAR_SECOND : 초
- I18N_UCALENDAR_MILLISECOND : 밀리세컨

i18n_udate : double 과 동일한 데이터 타입. posix 시간을 저장할때 사용합니다.

int i18n_ucalendar_get_milliseconds(i18n_ucalendar_h calendar,
i18n_udate *date) : i18n_ucalendar_h에 저장된 시간을 posix 시간으로
변경하는 API. 단위는 밀리세컨.

int i18n_ucalendar_add (i18n_ucalendar_h calendar,
i18n_ucalendar_date_fields_e field, int32_t amount) : i18n_ucalendar_h
객체의 특정 항목에 숫자를 더하는 API. / 파라미터
- i18n_ucalendar_h 객체, 날짜&시간 필드, 더해지는 숫자.

29. Mouse Touch 이벤트 구하기

이미지 뷰어를 구현하려면 터치 드래그로 이미지가 슬라이드 전환되는 기능을 구현해야 합니다. 오카리나, 피아노, 악기 같은 악기 앱은 멀티터치 정보를 구해야 합니다. 그 외에도 대부분의 게임을 비롯해서 수준있는 앱들은 터치 이벤트를 사용합니다. 이번 시간에는 사용자가 윈도우를 터치했을 때의 이벤트를 구하는 방법을 알아보겠습니다.

1) 컨테이너 Touch 이벤트 구하기

새로운 소스 프로젝트를 생성하고 Project name을 MouseTouchEvent 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "Hello EFL");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);
evas_object_show(ad->label);

/* Mouse Touch event callback */
evas_object_event_callback_add( ad->conform, EVAS_CALLBACK_MOUSE_DOWN,
on_mouse_down , ad);
evas_object_event_callback_add( ad->conform, EVAS_CALLBACK_MOUSE_MOVE,
on_mouse_move , ad);
evas_object_event_callback_add( ad->conform, EVAS_CALLBACK_MOUSE_UP,
```

```
on_mouse_up , ad);

/* Show window after base gui is set up */
evas_object_show(ad->win);
```

evas_object_event_callback_add() 은 evas 객체에 콜백 함수를 지정하는 API입니다. evas 객체는 화면에 보여지는 모든 객체를 의미합니다.

따라서 기본 객체(Line, Rect, Polygon, Text, Image)와 스마트 객체(컨테이너, 위젯)를 모두 포함합니다.

1번째 파라미터에는 이벤트가 발생하는 객체를 지정합니다. 여기서는 Conformant를 지정하였습니다.

2번째 파라미터에는 이벤트의 종류를 지정합니다.

EVAS_CALLBACK_MOUSE_DOWN 는 Touch 다운 이벤트를 의미합니다.

EVAS_CALLBACK_MOUSE_MOVE 는 Touch 이동 이벤트를 의미합니다.

EVAS_CALLBACK_MOUSE_UP 는 Touch 해제 이벤트를 의미합니다.

3번째 파라미터는 콜백 함수명을 지정하고, 4번째는 사용자 데이터입니다.

사용자가 Conformant를 Touch 했을 때 호출되는 콜백 함수를 만들어 보겠습니다. create_base_gui() 함수 위에 새로운 함수 3개를 추가합니다.

```
static void
on_mouse_down(void *data, Evas *e, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    Evas_Event_Mouse_Down *ev = event_info;
    char buf[100];

    sprintf(buf, "Win Mouse down:%d,%d", ev->canvas.x, ev->canvas.y);
    elm_object_text_set(ad->label, buf);
```

```

}

static void
on_mouse_move(void *data, Evas *e, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    Evas_Event_Mouse_Move *ev = event_info;
    char buf[100];

    sprintf(buf, "Win Mouse move:%d,%d/%d,%d",
            ev->prev.canvas.x, ev->prev.canvas.y, ev->cur.canvas.x, ev->cur.canvas.y);
    elm_object_text_set(ad->label, buf);
}

static void
on_mouse_up(void *data, Evas *e, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    Evas_Event_Mouse_Up *ev = event_info;
    char buf[100];

    sprintf(buf, "Win Mouse up:%d,%d", ev->canvas.x, ev->canvas.y);
    elm_object_text_set(ad->label, buf);
}

```

`on_mouse_down()` 는 Conformant를 Touch 다운했을 때의 콜백 함수입니다. 1번째 파라미터에는 사용자 데이터가 전달되고, 2번째 파라미터에는 이벤트가 발생한 객체가 전달됩니다. 3번째 파라미터에는 Touch 이벤트 정보가 저장된 `Evas_Event_Mouse_Down` 객체가 전달됩니다.

`Evas_Event_Mouse_Down` 객체의 속성 중에서 canvas에는 Touch 포인트 좌표가 저장되어 있습니다.

`on_mouse_move()` 는 `Conformant`를 `Touch` 이동했을 때의 콜백 함수입니다. 1번째 파라미터에는 사용자 데이터가 전달되고, 2번째 파라미터에는 이벤트가 발생한 객체가 전달됩니다. 3번째 파라미터에는 `Touch` 이벤트 정보가 저장된 `Evas_Event_Mouse_Move` 객체가 전달됩니다.

`Evas_Event_Mouse_Move` 객체의 속성 중에서 `prev.canvas`에는 이전 `Touch` 포인트 좌표가 저장되어 있습니다. `cur.canvas`에는 현재 `Touch` 포인트 좌표가 저장되어 있습니다.

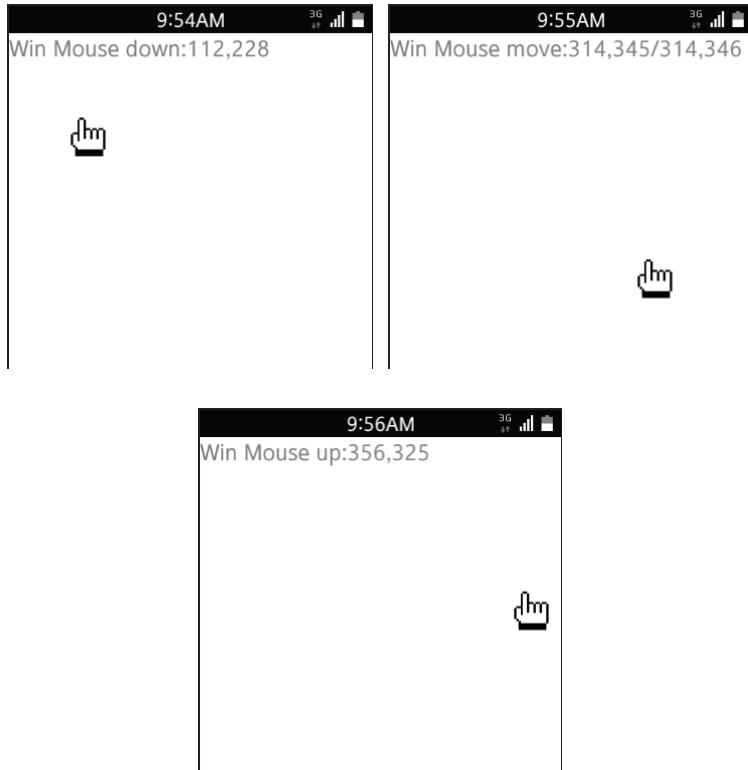
`on_mouse_up()` 은 `Conformant`를 `Touch` 해제했을 때의 콜백 함수입니다. 1번째 파라미터에는 사용자 데이터가 전달되고, 2번째 파라미터에는 이벤트가 발생한 객체가 전달됩니다. 3번째 파라미터에는 `Touch` 이벤트 정보가 저장된 `Evas_Event_Mouse_Up` 객체가 전달됩니다.

`Evas_Event_Mouse_Up` 객체의 속성 중에서 `canvas`에는 `Touch` 포인트 좌표가 저장되어 있습니다.

예제를 빌드하고 실행시켜 봅시다. 마우스로 화면을 `Touch`하면 포인트 좌표가 `Label` 위젯에 표시됩니다.

마우스를 드래그하면 이전 `Touch` 포인트 좌표와 현재 `Touch` 포인트 좌표가 `Label` 위젯에 표시됩니다.

마우스 `Touch`를 해제하면 마지막 포인트 좌표가 `Label` 위젯에 표시됩니다.



3) 멀티 Touch 이벤트 구하기

이번에는 여러개의 손가락으로 터치 했을 때 각각의 Touch 정보를 구해보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* Mouse Touch event callback */
evas_object_event_callback_add(    ad->conform,    EVAS_CALLBACK_MOUSE_DOWN,
on_mouse_down , ad);
evas_object_event_callback_add(    ad->conform,    EVAS_CALLBACK_MOUSE_MOVE,
on_mouse_move , ad);
evas_object_event_callback_add(    ad->conform,    EVAS_CALLBACK_MOUSE_UP,
on_mouse_up , ad);
```

```
/* Multi Touch event callback */
```

```

evas_object_event_callback_add(ad->conform,    EVAS_CALLBACK_MULTI_DOWN,
multi_down_cb, ad);
evas_object_event_callback_add(ad->conform,    EVAS_CALLBACK_MULTI_MOVE,
multi_move_cb, ad);

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

EVAS_CALLBACK_MULTI_DOWN 는 멀티 Touch 다운 이벤트를 의미합니다.

EVAS_CALLBACK_MULTI_MOVE 는 멀티 Touch 이동 이벤트를 의미합니다.

멀티 Touch 이벤트 함수를 만들 차례입니다. create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

```

static void
multi_down_cb(void *data, Evas *e, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    Evas_Event_Multi_Down *ev = (Evas_Event_Multi_Down*)event_info;
    char buf[100];

    sprintf(buf, "Multi down : %d - %d,%d", ev->device, ev->canvas.x, ev->canvas.y);
    elm_object_text_set(ad->label, buf);
}

static void
multi_move_cb(void *data, Evas *e, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

```

```

Evas_Event_Multi_Move *ev = (Evas_Event_Multi_Move*)event_info;
char buf[100];

sprintf(buf, "Multi move : %d - %d,%d", ev->device, ev->cur.canvas.x, ev->cur.canvas.y);
elm_object_text_set(ad->label, buf);

}

```

multi_down_cb() 는 멀티 Touch 다운 이벤트 함수입니다. 1번째 포인트의 이벤트는 수신되지 않습니다. 3번째 파라미터에는 Touch 이벤트 정보가 저장된 Evas_Event_Multi_Down 객체가 전달됩니다.

Evas_Event_Multi_Down 객체의 속성 중에서 canvas에는 Touch 포인트 좌표가 저장되어 있습니다. 속성 중에서 device에는 해당 포인트의 ID가 저장되어 있습니다. Move 이벤트 혹은 Up 이벤트가 발생하면 ID로 구분할 수 있습니다.

multi_move_cb() 는 멀티 Touch 이동 이벤트 함수입니다. 1번째 포인트의 이벤트는 수신되지 않습니다. 3번째 파라미터에는 Touch 이벤트 정보가 저장된 Evas_Event_Multi_Move 객체가 전달됩니다.

Evas_Event_Multi_Move 객체의 속성 중에서 cur에는 Touch 포인트 좌표가 저장되어 있습니다. 속성 중에서 device에는 해당 포인트의 ID가 저장되어 있습니다. Down 이벤트가 발생했을 때의 ID로 구분할 수 있습니다.

예제를 다시 실행합니다. 에뮬레이터에서 멀티 Touch를 테스트하려면 키보드 Ctrl 키를 계속 누르고 있는 상태에서 마우스 왼쪽 버튼으로 2군데를 터치하면 됩니다. 그러면 회색 동그라미가 표시됩니다. 멀티 Touch 이동 이벤트를 테스트 하려면 회색 동그라미를 드래그하면 됩니다. Ctrl 키를 해제하면 멀티 Touch 표시도 사라집니다.



4) 관련 API

void evas_object_event_callback_add(Evas_Object *obj, Evas_Callback_Type type, Evas_Object_Event_Cb func, void *data) : evas 객체에 콜백 함수를 지정하는 API. evas 객체는 화면에 보여지는 모든 객체를 의미합니다. 따라서 기본 객체(Line, Rect, Polygon, Text, Image)와 스마트 객체(컨테이너, 위젯)를 모두 포함합니다. / 파라미터 : 이벤트가 발생하는 객체, 이벤트의 종류, 콜백 함수명, 사용자 데이터. 이벤트 종류는 다음과 같습니다.

- EVAS_CALLBACK_MOUSE_DOWN : Touch 다운 이벤트
- EVAS_CALLBACK_MOUSE_MOVE : Touch 이동 이벤트
- EVAS_CALLBACK_MOUSE_UP : Touch 해제 이벤트
- EVAS_CALLBACK_MULTI_DOWN : 멀티 Touch 다운 이벤트
- EVAS_CALLBACK_MULTI_MOVE : 멀티 Touch 이동 이벤트

Evas_Event_Mouse_Down : Touch 다운 이벤트 정보 구조체. 속성 중에서 canvas에는 Touch 포인트 좌표가 저장되어 있습니다.

Evas_Event_Mouse_Move : Touch 이동 이벤트 정보 구조체. 속성 중에서 prev.canvas에는 이전 Touch 포인트 좌표가 저장되어 있습니다. cur.canvas에는 현재 Touch 포인트 좌표가 저장되어 있습니다.

Evas_Event_Mouse_Up : Touch 해제 이벤트 정보 구조체. 속성 중에서 canvas에는 Touch 포인트 좌표가 저장되어 있습니다.

Evas_Event_Multi_Down : 멀티 Touch 다운 이벤트 정보 구조체. 속성 중에서 canvas에는 Touch 포인트 좌표가 저장되어 있습니다. 속성 중에서 device에는 해당 포인트의 ID가 저장되어 있습니다. Move 이벤트 혹은 Up 이벤트가 발생하면 ID로 구분할 수 있습니다.

Evas_Event_Multi_Move : Touch 이동 이벤트 정보 구조체. 객체의 속성 중에서 cur에는 Touch 포인트 좌표가 저장되어 있습니다. 속성 중에서 device에는 해당 포인트의 ID가 저장되어 있습니다. Down 이벤트가 발생했을 때의 ID로 구분할 수 있습니다.

30. 계산기 예제

이번 시간에는 수학 함수를 사용해서 간단한 계산기를 만드는 방법을 알아보겠습니다.

1) 화면 UI 구성

새로운 소스 프로젝트를 생성하고 Project name을 CalculatorEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 appdata 구조체에 새로운 변수를 추가하고 라이브러리 헤더파일과 define 정의도 추가합니다.

```
#include "calculatorex.h"
#include <math.h>

#define ID_BACK          101
#define ID_CLEAR         102
#define ID_DOT           103
#define ID_EQUAL          104
#define ID_PLUS          111
#define ID_MINUS         112
#define ID_MULTIPLY      113
#define ID_DIVIDE        114
#define ID_X2            121
#define ID_X3            122
#define ID_SQRT          123
#define ID_RECIPE         124
```

```
typedef struct appdata {
```

```
Evas_Object *win;  
Evas_Object *conform;  
Evas_Object *entry;  
float value;  
int calc_mode;  
} appdata_s;
```

math.h는 수학함수 라이브러리 헤더파일입니다.

define 문에는 Button을 구분하기 위한 ID를 정의합니다.

appdata 구조체에 추가된 entry에는 계산 결과값을 표시하고, value에는 임시 계산 결과값을 저장합니다.

calc_mode에는 연산의 종류(+, -, *, /)를 저장합니다.

이번 예제에서는 많은 숫자의 Button 위젯을 생성하게 됩니다.
소스코드를 줄이기 위해서 Button 생성 함수를 추가하겠습니다.
create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

```
static void  
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)  
{  
}  
  
static void  
create_button(Evas_Object *parent, const char* text, int x, int y, int w, int h, void *data)  
{  
    Evas_Object *btn = elm_button_add(parent);  
    elm_object_text_set(btn, text);
```

```

    evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, EVAS_HINT_FILL);
    elm_table_pack(parent, btn, x, y, w, h);
    evas_object_smart_callback_add(btn, "clicked", btn_clicked_cb, data);
    evas_object_show(btn);
}


```

btn_clicked_cb() 는 Button 콜백 함수입니다. 여러개의 Button이 동일한 콜백 함수를 호출하게 됩니다.

create_button() 은 Button 관련 정보를 받아서 Button 위젯을 생성하는 함수입니다.

Box와 Table 그리고 22개의 Button을 생성하겠습니다. create_base_gui() 함수에 새로운 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* Box to put the table in so we can bottom-align the table
 * window will stretch all resize object content to win size */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, box);


```

```

evas_object_show(box);

/* Table */
Evas_Object *table = elm_table_add(ad->win);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 10 * elm_config_scale_get(), 10 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fiill width but align to bottom of box */
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, 1.0);
elm_object_content_set(ad->conform, table);
evas_object_show(table);

{ /* child object - indent to how relationship */
/* Entry */
ad->entry = elm_entry_add(ad->win);
elm_object_text_set(ad->entry, "0");
evas_object_size_hint_weight_set(ad->entry, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(ad->entry, EVAS_HINT_FILL,
EVAS_HINT_FILL);
elm_table_pack(table, ad->entry, 0, 0, 4, 1);
evas_object_show(ad->entry);

create_button(table, "Back", 0, 1, 2, 1, ID_BACK);
create_button(table, "Clear", 2, 1, 2, 1, ID_CLEAR);

create_button(table, "7", 0, 2, 1, 1, 7);
create_button(table, "8", 1, 2, 1, 1, 8);
create_button(table, "9", 2, 2, 1, 1, 9);
create_button(table, "/", 3, 2, 1, 1, ID_DIVIDE);

```

```

create_button(table, "4", 0, 3, 1, 1, 4);
create_button(table, "5", 1, 3, 1, 1, 5);
create_button(table, "6", 2, 3, 1, 1, 6);
create_button(table, "*", 3, 3, 1, 1, ID_MULTIPLY);

create_button(table, "1", 0, 4, 1, 1, 1);
create_button(table, "2", 1, 4, 1, 1, 2);
create_button(table, "3", 2, 4, 1, 1, 3);
create_button(table, "-", 3, 4, 1, 1, ID_MINUS);

create_button(table, "0", 0, 5, 1, 1, 0);
create_button(table, ".", 1, 5, 1, 1, ID_DOT);
create_button(table, "=", 2, 5, 1, 1, ID_EQUAL);
create_button(table, "+", 3, 5, 1, 1, ID_PLUS);

create_button(table, "x^2", 0, 6, 1, 1, ID_X2);
create_button(table, "x^3", 1, 6, 1, 1, ID_X3);
create_button(table, "sqrt", 2, 6, 1, 1, ID_SQRT);
create_button(table, "1/x", 3, 6, 1, 1, ID_RECIPE);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

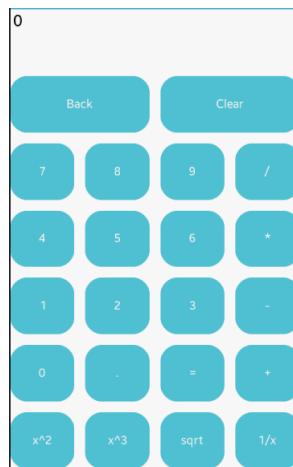
위젯들을 비율 단위로 화면에 배치하기 위해서 Table을 생성하였고, 위젯 사이에 간격을 지정하기 위해서 Box를 생성하였습니다.

계산 결과값을 표시하기 위해서 Entry 위젯을 생성하였습니다.

그 다음은 총 22개의 Button 위젯을 생성하는 코드입니다. 숫자 Button 인 경우에는 콜백 함수에 해당 숫자값을 사용자 데이터로 전달합니다.

그외 Button 인 경우에는 define 문에서 정의한 ID 값을 사용자 데이터로 전달합니다.

예제를 빌드하고 실행시켜 볼시다. 화면 위쪽에 Entry 위젯이 있고 그 아래에 22개의 Button 위젯이 생성되었습니다.



2) 숫자 Button 기능 구현

0~9까지의 숫자 Button을 클릭하면 Label 위젯에 해당 숫자가 추가되는 기능을 구현해 보겠습니다. 우선 appdata를 전역변수로 선언합니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    Evas_Object *nf;  
    Elm_Object_Item *frame_item;  
    float value;  
    int calc_mode;  
} appdata_s;
```

```
appdata_s* m_ad = 0;
```

```
_____
```

그런 다음 `create_base_gui()` 함수 시작 부분에서 `appdata` 변수를 초기화 합니다.

```
static void  
create_base_gui(appdata_s *ad)  
{  
    m_ad = ad;
```

```
_____
```

`Entry`의 캡션 텍스트를 숫자로 변경하는 함수와 `Entry` 위젯에 글자를 추가하는 함수를 생성하겠습니다. `btn_clicked_cb()` 함수 위에 새로운 함수 2개를 추가합니다.

```
static float  
get_entry_value()  
{  
    char* text = elm_object_text_get(m_ad->entry);  
    float value = atof(text);  
    return value;  
}
```

```
static void  
append_number_label(char str_new) {  
    char buf[100];  
  
    char* text = elm_object_text_get(m_ad->entry);  
    float value = get_entry_value();  
    if( value == 0.f )
```

```
    sprintf(buf, "%c", str_new);
else
    sprintf(buf, "%s%c", text, str_new);

elm_object_text_set(m_ad->entry, buf);
}
```

get_entry_value() 는 entry 캡션 텍스트를 float 형식으로 변경해서 반환하는 함수입니다.

append_number_label() 는 char 변수를 받아서 entry 캡션 텍스트 끝에 추가하는 함수입니다.

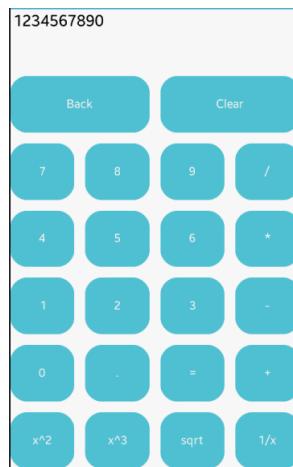
사용자가 숫자 Button을 클릭하면 위 함수를 호출해주면 됩니다.
btn_clicked_cb() 함수에 다음 코드를 추가합니다.

```
static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{
    char* text = NULL;
    int length = 0;
    float value = 0.f;
    int id = (int)data;

    if( id >= 0 && id <= 9 )
    {
        append_number_label('0' + id);
        return;
    }
}
```

사용자 데이터가 0~9 사이인 경우에는 숫자 Button 으로 간주하고 Label 위젯의 캡션 텍스트에 새로운 문자를 추가합니다.

예제를 다시 실행시켜 봅시다. 그리고 숫자 Button을 클릭하면 Label 위젯에 해당 숫자가 추가됩니다.



3) Dot, Clear, Back Button 기능 구현

'.' 버튼, Back 버튼, 그리고 Clear 버튼에 기능을 구현해 보겠습니다. btn_clicked_cb() 함수 끝부분에 새로운 코드를 추가합니다.

```
if( id >= 0 && id <= 9 )
{
    append_number_label('0' + id);
    return;
}

switch( id )
{
    case ID_DOT :
```

```

append_number_label('.');
break;

case ID_CLEAR :
    elm_object_text_set(m_ad->label, "0");
    break;

case ID_BACK :
    text = elm_object_text_get(m_ad->label);
    length = strlen(text);
    if( length > 0 )
        text = eina_stringshare_add_length(text, length - 1);
    if( strlen(text) < 1 )
        text = "0";
    elm_object_text_set(m_ad->label, text);
    break;

}

}

```

사용자가 '.' Button을 클릭했다면 Label 텍스트 끝에 '.' 기호를 추가합니다.

사용자가 Clear Button을 클릭했다면 Label 텍스트를 '0' 으로 변경합니다.

사용자가 Back Button을 클릭했다면 Label 텍스트 끝 문자를 삭제합니다.

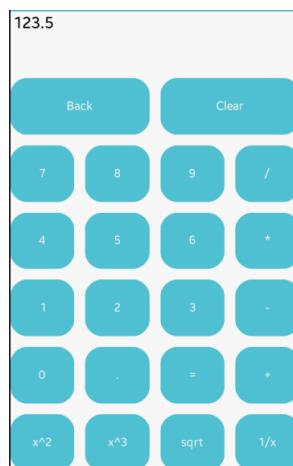
eina_stringshare_add_length(char*, unsigned int) 는 길이를 지정해서 문자열 앞부분을 추출해 주는 API 입니다.

예제를 다시 실행시켜 봅시다. 숫자 '.' Button을 누르면 Label 텍스트

오른쪽 끝에 '.' 기호가 추가됩니다.

Back Button을 누르면 오른쪽 한 문자가 삭제됩니다.

Clear Button을 누르면 텍스트 전체가 0으로 변경됩니다.



4) 사칙연산

float 형식의 값을 Label 위젯에 입력하고, '=' Button을 눌렀을 때 연산을 수행하는 함수를 생성하겠습니다. btn_clicked_cb() 함수 위에 새로운 함수 2개를 추가합니다.

```
static void  
set_entry_value(float value)  
{  
    char buf[100];  
    sprintf(buf, "%f", value);  
    elm_object_text_set(m_ad->entry, buf);  
}
```

```
static void
btn_equal_clicked()
{
    float value2 = get_entry_value();

    switch( m_ad->calc_mode )
    {
        case ID_PLUS :
            m_ad->value += value2;
            break;
        case ID_MINUS :
            m_ad->value -= value2;
            break;
        case ID_MULTIPLY :
            m_ad->value *= value2;
            break;
        case ID_DIVIDE :
            m_ad->value /= value2;
            break;
    }

    set_entry_value(m_ad->value);
}
```

set_label_value() 는 실수값을 받아서 문자열로 변환한 다음 Label 위젯에 입력하는 함수입니다.

btn_equal_clicked() 는 연칙연산 종류에 따라서 2개의 숫자를 계산하는 함수입니다.

btn_clicked_cb() 함수에 새로운 코드를 추가합니다.

```

~  

case ID_BACK :  

    text = elm_object_text_get(m_ad->label);  

    length = strlen(text);  

    if( length > 0 )  

        text = eina_stringshare_add_length(text, length - 1);  

    if( strlen(text) < 1 )  

        text = "0";  

    elm_object_text_set(m_ad->label, text);  

    break;  

case ID_PLUS :  

case ID_MINUS :  

case ID_MULTIPLY :  

case ID_DIVIDE :  

    m_ad->value = get_label_value();  

    elm_object_text_set(m_ad->label, "0");  

    m_ad->calc_mode = id;  

    break;  

case ID_EQUAL :  

    btn_equal_clicked();  

    break;  

}  

}

```

사용자가 사칙연산 Button을 누르면 Entry 텍스트를 숫자로 변경해서 전역변수에 저장합니다.

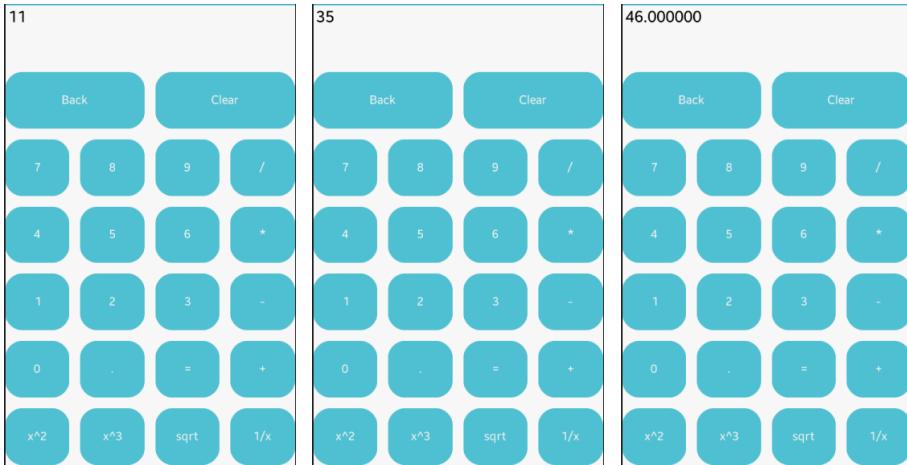
Entry 텍스트는 '0'으로 변경하고 사칙연산 종류를 전역변수에 저장해둡니다.

사용자가 '=' Button을 클릭하면 btn_equal_clicked() 함수를 호출합니다.

예제를 다시 실행하고 '11'을 입력했다가 '+' Button을 누르면 Entry 위젯에 '0'이 표시됩니다.

그런 다음 '35'를 입력했다가 '=' Button을 누르면 Entry 위젯에 연산 결과가 표시됩니다.

뺄셈, 곱셈, 나눗셈도 테스트 해보기 바랍니다.



5) 제곱, 제곱근 구하기

수학 함수를 사용해서 제곱과 제곱근 값을 구해 보겠습니다.

btn_clicked_cb() 함수 끝부분에 새로운 코드를 추가합니다.

```
~  
case ID_EQUAL :  
    btn_equal_clicked();  
    break;  
case ID_X2 :  
    value = get_label_value();  
    value = pow( value, 2 );  
    set_label_value(value);  
    break;  
case ID_X3 :  
    value = get_label_value();
```

```
    value = pow( value, 3 );
    set_label_value(value);
    break;

case ID_SQRT :
    value = get_label_value();
    value = sqrt( value );
    set_label_value(value);
    break;

case ID_RECIPE :
    value = get_label_value();
    value = 1.f / value;
    set_label_value(value);
    break;

}

}
```



`pow(double, double)` 은 제곱값을 구하는 수학함수입니다. 2번째 파라미터에 2를 전달하면 제곱값을 반환하고, 3을 전달하면 3제곱값을 반환합니다.

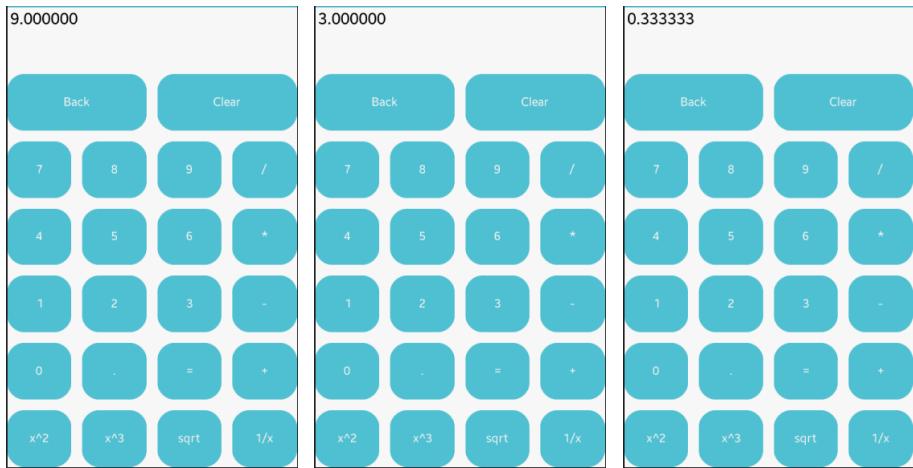
`sqrt(double)` 는 제곱근을 반구하는 수학함수입니다.

역수를 구할 때는 1을 숫자로 나눠주면 됩니다.

예제를 다시 실행시켜 봅시다. 숫자 3을 입력한 다음 ' x^2 ' Button을 누르면 제곱값이 표시됩니다.

그 상태에서 'sqrt' Button을 누르면 제곱근이 구해지기 때문에 원래 값이 표시됩니다.

그 상태에서 ' $1/x$ ' Button을 누르면 역수가 표시됩니다.



6) 관련 API

`double pow(double, double)` : 제곱값을 구하는 수학함수 API. /
파라미터 - 원본 숫자, 제곱 회수.

`double sqrt(double)` : 제곱근을 반구하는 수학함수 API. / 파라미터 -
원본 숫자.

31. 캔버스에 그라데이션 출력

'앱 개발을 잘하려면 어떤 능력이 뛰어나야 하는가?'

제가 강의를 할때 종종 이런 질문을 접하는데 그때 마다 저는 주저없이 이렇게 대답합니다.

"첫째는 그래픽을 잘 표현하는 능력,

둘째는 OOP (객체지향 프로그래밍)를 잘 사용하는 능력이 가장 중요합니다."

시간이 지날수록 개발은 더 쉬워지고 있습니다. 예전에는 개발자가 직접 하드코딩으로 기능을 구현하는 것이 일반적이었습니다. 하지만 이제는 플랫폼에서 많은 기능을 제공하고 있으며 개발자는 플랫폼에서 제공하는 API를 호출해서 사용하기만 하면 되는 것입니다.

상용 스마트폰 앱을 개발하다 보면 가장 힘들고 손이 많이 가는 부분이 그래픽입니다. 2가지 어플리케이션이 동일한 기능을 가지고 있을 때 사용자는 그래픽이 더 화려한 쪽을 선호합니다. 사용자의 취향을 고려하여 더 다양하고 화려한 그래픽을 구현하는 기술이 개발자가 갖추어야 할 가장 중요한 능력입니다.

Evas는 EFL에서 제공하는 캔버스입니다. Evas에 그린 모든 도형은 객체로 생성됩니다. 이번 시간에는 캔버스에 라인을 그리는 방법을 알아보겠습니다.

1) 캔버스 생성 & 그라데이션 그리기

새로운 소스 프로젝트를 생성하고 Project name을 DrawGradiation 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 위 부분에 아래와 같이 새로운 코드를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    //Evas_Object *label;
    Evas_Object *imgs[5];
} appdata_s;

static Evas_Object *
create_gradient_rect(appdata_s *ad)
{
    /* Generate gradient data on the fly */
    const int colors[2][4] = {
        /* red to blue */
        { 255, 0, 0, 255 }, { 0, 0, 255, 255 },
    };

    const int b_r = colors[0][0], b_g = colors[0][1], b_b = colors[0][2], b_a =
colors[0][3];
    const int e_r = colors[1][0], e_g = colors[1][1], e_b = colors[1][2], e_a =
colors[1][3];

    Evas_Object *img;
    unsigned int *data32;

    /* Create image object, set its image data size & type */
    Evas* canvas = evas_object_evas_get(ad->win);
    img = evas_object_image_filled_add(canvas);
```

```

/* BGRA data */
evas_object_image_colorspace_set(img, EVAS_COLORSPACE_ARGB8888);
/* Size is 255x1 */
evas_object_image_size_set(img, 255, 1);
/* Mark image as having alpha */
evas_object_image_alpha_set(img, EINA_TRUE);

/* get a writable data pointer */
data32 = evas_object_image_data_get(img, EINA_TRUE);

for (unsigned x = 0; x < 255; x++)
{
    int r, g, b, a;
    /* interpolate alpha */
    a = (b_a * (255 - x) + e_a * x) / (2 * 255);
    /* interpolate red */
    r = (b_r * b_a * (255 - x) + e_r * e_a * (x)) / (2 * 255 * 255);
    /* interpolate green */
    g = (b_g * b_a * (255 - x) + e_g * e_a * (x)) / (2 * 255 * 255);
    /* interpolate blue */
    b = (b_b * b_a * (255 - x) + e_b * e_a * (x)) / (2 * 255 * 255);
    /* write pixel value now */
    data32[x] = (a << 24) | (r << 16) | (g << 8) | b;
}

/* very important: set data back */
evas_object_image_data_set(img, data32);

evas_object_size_hint_weight_set(img, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(img, EVAS_HINT_FILL, EVAS_HINT_FILL);
evas_object_show(img);
return img;
}

```

화면에 그림을 그리려면 Evas 라는 캔버스를 사용하고, 이번 예제에서는 총 5개의 Image 객체를 생성해보겠습니다. appdata 구조체에 추가된 imgs[5] 는 Image 객체를 저장하는 배열 변수입니다.

create_gradient_rect() 는 그라데이션 Image 객체를 생성해서 반환하는 함수입니다. 함수의 내용을 하나씩 설명 드리겠습니다.

colors[2][4]에는 2가지 컬러를 저장하는 배열 변수입니다. 1번째는 그라데이션 시작 컬러이고, 2번째는 그라데이션 종료 컬러입니다. 컬러는 총 4개의 숫자값으로 이루어집니다. 순서대로 Red, Green, Blue, Alpha(반투명) 입니다.

evas_object_evas_get() 는 Evas 객체를 생성하는 API 입니다.

evas_object_image_filled_add() 는 Image 객체를 생성하는 API 입니다.

evas_object_image_colorspace_set() 는 Image 객체의 색공간을 지정하는 API 입니다. 2번째 파라미터에 EVAS_COLORSPACE_ARGB8888 를 전달하면 하나의 픽셀이 4개의 데이터(Red, Green, Blue, Alpha)로 구성됩니다.

evas_object_image_size_set() 는 Image 객체의 크기를 지정하는 API 입니다. 2번째 파라미터에 255를 전달하면 수평 픽셀 개수가 255개 입니다. 3번째 파라미터에 1을 전달하면 수직 픽셀 개수가 1개 입니다.

evas_object_image_alpha_set() 는 Image 객체에 반투명 적용여부를 지정하는 API 입니다.

evas_object_image_data_get() 는 Image 객체의 원본 데이터를 배열로 반환하는 API 입니다.

`evas_object_image_data_set()` 는 `evas_object_image_data_get()` 와 반대 역할을 합니다. Image 객체에 원본 데이터를 지정하는 API 입니다.

이 함수를 이용해서 그라데이션 Image 객체를 생성해서 화면에 출력해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다. Label은 필요없으니 주석처리 합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

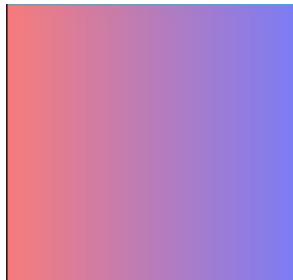
Evas_Object *box = elm_box_add(ad->conform);
elm_box_padding_set(box, ELM_SCALE_SIZE(10), ELM_SCALE_SIZE(10));
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    ad->imgs[0] = create_gradient_rect(ad);
    elm_box_pack_end(box, ad->imgs[0]);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);
```

Box와 Image 객체를 생성해서 화면에 출력하였습니다. 예제를 실행시켜 봅시다. 왼쪽은 빨간색이고 오른쪽은 파란색인 그라데이션 사각형이

출력되었습니다.



2) 다양한 그라데이션 사각형 생성

앞서 만들었던 `create_gradient_rect()` 함수에 4가지 종류의 그라데이션 사각형을 생성하는 기능을 구현해 보겠습니다. 아래와 같이 코드를 수정합니다.

```
static Evas_Object *
create_gradient_rect(appdata_s *ad, unsigned i)
//create_gradient_rect(appdata_s *ad)
{
    /* Generate gradient data on the fly */
    const int colors[8][4] = {
        /* red to blue */
        { 255, 0, 0, 255 }, { 0, 0, 255, 255 },
        /* black to transparent */
        { 0, 0, 0, 255 }, { 0, 0, 0, 0 },
        /* green to orange */
        { 0, 255, 0, 255 }, { 255, 128, 0, 255 },
        /* yellow to cyan */
        { 255, 255, 0, 255 }, { 0, 255, 255, 255 }
    };
    /*const int colors[2][4] = {
```

```

red to blue
{ 255, 0, 0, 255 }, { 0, 0, 255, 255 },
};

const int b_r = colors[i*2][0], b_g = colors[i*2][1], b_b = colors[i*2][2], b_a =
colors[i*2][3];
const int e_r = colors[i*2+1][0], e_g = colors[i*2+1][1], e_b = colors[i*2+1][2],
e_a = colors[i*2+1][3];

Evas_Object *img;
unsigned int *data32;

```

이제 위 함수를 4번 호출해 주면 4가지 그라데이션 Image를 생성할 수 있습니다. `create_base_gui()` 함수에 아래 부분을 아래와 같이 수정합니다.

```

{
    //ad->imgs[0] = create_gradient_rect(ad);
    //elm_box_pack_end(box, ad->imgs[0]);
    for (unsigned i = 0; i < 4; i++)
    {
        ad->imgs[i] = create_gradient_rect(ad, i);
        elm_box_pack_end(box, ad->imgs[i]);
    }
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

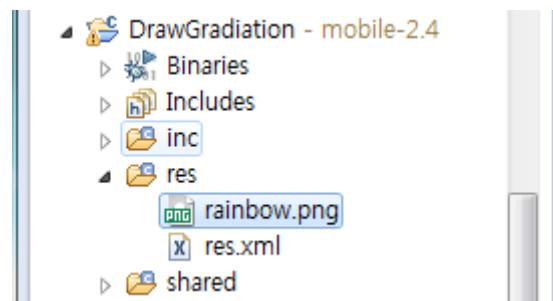
```

예제를 다시 실행시켜 봅시다. 4개의 그라데이션 사각형이 출력됩니다.



3) Image 파일을 이용한 무지개 사각형 생성

이번에는 Image 파일을 사용해서 무지개 사각형을 화면에 출력해 보겠습니다. 부록 /image 폴더에 있는 rainbow.png 파일을 소스 프로젝트 /res 폴더로 복사합니다.



이제 소스코드에서 Image 파일을 로딩해서 Image 객체에 지정하면 됩니다. 소스파일로 이동해서 `create_base_gui()` 함수 위에 새로운 함수를 생성합니다.

```
static Evas_Object *
create_rainbow_rect(appdata_s *ad)
{
```

```

/* A much simpler method for gradients is to simply use an image from disk */
Evas_Object *img;
char path[PATH_MAX];

/* Create image object, set its image data size & type */
Evas* canvas = evas_object_evas_get(ad->win);
img = evas_object_image_filled_add(evas_object_evas_get(canvas));

snprintf(path, sizeof(path), "%s/rainbow.png", app_get_resource_path());
dlog_print(DLOG_ERROR, LOG_TAG, "path: '%s'", path);
evas_object_image_file_set(img, path, NULL);

evas_object_size_hint_weight_set(img, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(img, EVAS_HINT_FILL, EVAS_HINT_FILL);
evas_object_show(img);
return img;
}

```

evas_object_image_file_set() 는 Image 객체에 파일 경로를 지정해서 파일을 로딩하는 API입니다.

이제 위 함수를 사용해서 이미지 파일을 로딩해서 화면에 출력해 보겠습니다. create_base_gui() 함수에 새로운 코드를 추가합니다.

```

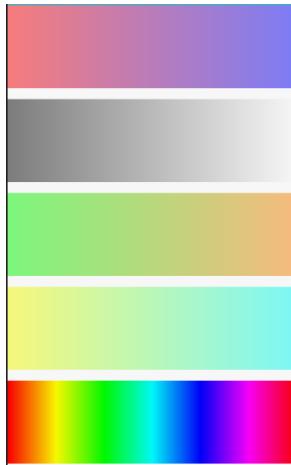
{
    for (unsigned i = 0; i < 4; i++)
    {
        ad->imgs[i] = create_gradient_rect(ad, i);
        elm_box_pack_end(box, ad->imgs[i]);
    }

    ad->imgs[5] = create_rainbow_rect(ad);
}

```

```
elm_box_pack_end(box, ad->imgs[5]);  
}  
_____
```

예제를 다시 실행하면 화면 아래쪽에 무지개 사각형이 출력됩니다.



4) 관련 API

Evas *evas_object_evas_get(Evas_Object *obj) : Evas 객체를 생성하는 API.
/ 파라미터 - 원도우 객체.

evas_object_image_filled_add() 는 Image 객체를 생성하는 API 입니다.

evas_object_image_colorspace_set() 는 Image 객체의 색공간을 지정하는 API 입니다. 2번째 파라미터에 EVAS_COLORSPACE_ARGB8888 를 전달하면 하나의 픽셀이 4개의 데이터(Red, Green, Blue, Alpha)로 구성됩니다.

evas_object_image_size_set() 는 Image 객체의 크기를 지정하는 API 입니다. 2번째 파라미터에 255를 전달하면 수평 픽셀 개수가 255개

입니다. 3번째 파라미터에 1을 전달하면 수직 픽셀 개수가 1개 입니다.

evas_object_image_alpha_set() 는 Image 객체에 반투명 적용여부를 지정하는 API 입니다.

evas_object_image_data_get() 는 Image 객체의 원본 데이터를 배열로 반환하는 API 입니다.

evas_object_image_data_set() 는 evas_object_image_data_get() 와 반대 역할을 합니다. Image 객체에 원본 데이터를 지정하는 API 입니다.

evas_object_image_file_set() 는 Image 객체에 파일 경로를 지정해서 파일을 로딩하는 API 입니다.

32. 캔버스에 사각형 출력

화면에 도형을 그릴 때는 캔버스를 사용하면 됩니다. Evas는 EFL에서 제공하는 캔버스입니다. Evas에 그린 모든 도형은 객체로 생성됩니다. 이번 시간에는 캔버스에 사각형 도형을 그리는 방법을 알아보겠습니다.

1) 캔버스 생성 & 사각형 그리기

새로운 소스 프로젝트를 생성하고 Project name을 DrawRect 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수에 새로운 코드를 추가합니다. Label은 필요없으니 주석처리 합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);*/

{ /* child object - indent to how relationship */
```

```

/* A grid to stretch content within grid size */
Evas_Object *grid = elm_grid_add(ad->win);
evas_object_size_hint_weight_set(grid, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, grid);
evas_object_show(grid);

{
    /* Canvas */
    Evas* canvas = evas_object_evas_get(ad->win);

    /* Rect-1 */
    Evas_Object *rect = evas_object_rectangle_add(canvas);
    evas_object_color_set(rect, 255, 0, 0, 192);
    evas_object_show(rect);
    elm_grid_pack(grid, rect, 4, 5, 52, 31);
}
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

elm_grid_add() 는 Grid 컨테이너를 생성하는 API 입니다. Grid는 오브젝트를 비율 단위로 화면에 배치할수 있는 컨테이너 입니다. Table은 셀 단위로 영역을 구분하고, Grid는 기본적으로 Max 값이 100으로 지정되어 있다는 것이 다릅니다.

elm_grid_pack() 은 Grid 컨테이너에 오브젝트를 배치하는 API 입니다. 1번째 파라미터는 Grid, 2번째는 오브젝트, 3번째는 수평 위치, 4번째는 수직 위치, 5번째는 수평 넓이, 6번째는 수직 넓이 입니다. 3번째 파라미터에 4를 전달하면 수평 방향으로 4% 위치를 의미합니다.

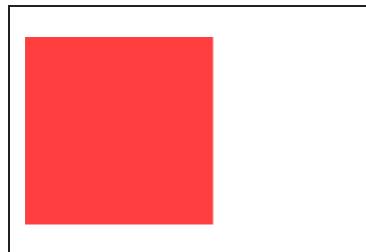
`elm_grid_size_set(obj, w, h)` 은 Grid 사이즈 수치를 지정하는 API 입니다.
기본 값은 수평으로 100, 수직으로 100 입니다.

`evas_object_evas_get(Evas_Object *)` 는 Evas 객체를 생성하는 API입니다.

`evas_object_rectangle_add(Evas *)` 는 캔버스에 Rectangle 객체를
생성하는 API입니다.

`evas_object_color_set(Evas_Object *, int, int, int, int)` 는 도형에 컬러를
지정하는 API입니다. 파라미터는 순서대로 Red, Green, Blue,
반투명도입니다. 255, 0, 0, 192로 지정하면 반투명이 적용된 빨간색
컬러가 만들어 집니다.

예제를 빌드하고 실행시켜 봅시다. 분홍색 사각형이 화면에
표시되었습니다. 빨간색을 지정했지만 반투명이 적용되어서 분홍색 처럼
보이는 것입니다.



2) 반투명 사각형 중첩

2개의 사각형을 추가해서 영역이 겹칠때 컬러가 어떻게 달라지는지 확인해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
[REDACTED]
```

```
/* Rect-1 */
Evas_Object *rect = evas_object_rectangle_add(canvas);
evas_object_color_set(rect, 255, 0, 0, 192);
evas_object_show(rect);
elm_grid_pack(grid, rect, 4, 5, 52, 31);

/* Rect-2 */
rect = evas_object_rectangle_add(canvas);
evas_object_color_set(rect, 0, 255, 0, 192);
evas_object_show(rect);
elm_grid_pack(grid, rect, 44, 5, 52, 31);

/* Rect-3 */
rect = evas_object_rectangle_add(canvas);
evas_object_color_set(rect, 0, 0, 255, 192);
evas_object_show(rect);
elm_grid_pack(grid, rect, 24, 29, 52, 31);

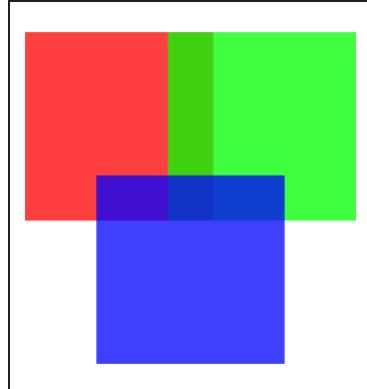
}
```

```
[REDACTED]
```

2번째 사각형의 컬러는 반투명이 적용된 Green으로 지정하였습니다.

3번째 사각형의 컬러는 반투명이 적용된 Blue로 지정하였습니다.

예제를 다시 실행시켜 봅시다. 3개의 사각형이 화면에 표시 되었습니다. 중첩된 부분은 중간 컬러로 바뀝니다. 반투명이 적용되었기 때문입니다.



3) 관련 API

Evas *evas_object_evas_get(Evas_Object *obj) : Evas 객체를 생성하는 API.
/ 파라미터 - 윈도우 객체.

Evas_Object *evas_object_rectangle_add(Evas *e) : 캔버스에 Rectangle 객체를 생성하는 API.

void evas_object_color_set(Evas_Object *obj, int r, int g, int b, int a) : 도형에 컬러를 지정하는 API. / 파라미터 - 도형 객체, Red 컬러, Green 컬러, Blue 컬러, 반투명도. 컬러값의 범위는 0~255 사이. 예를 들어서 노란색 컬러를 만들고 싶다면 255,255,0,255로 지정하면 됩니다.

33. 캔버스에 다각형 출력

화면에 도형을 그릴 때는 캔버스를 사용하면 됩니다. Evas 는 EFL에서 제공하는 캔버스입니다. Evas에 그린 모든 도형은 객체로 생성됩니다. 이번 시간에는 캔버스에 다각형 도형을 그리는 방법을 알아보겠습니다.

1) 캔버스 생성 & 삼각형 그리기

새로운 소스 프로젝트를 생성하고 Project name을 DrawPolygon으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수에 새로운 코드를 추가합니다. Label은 필요없으니 주석처리 합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);*/

{
```

```

/* Canvas */
Evas* canvas = evas_object_evas_get(ad->win);

/* Polygon - Triangle */
Evas_Object *polygon = evas_object_polygon_add(canvas);
evas_object_polygon_point_add(polygon, 20, 50);
evas_object_polygon_point_add(polygon, 170, 150);
evas_object_polygon_point_add(polygon, 20, 250);
evas_object_color_set(polygon, 255, 200, 0, 255);
evas_object_show(polygon);

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

evas_object_evas_get(Evas_Object *) 는 Evas 객체를 생성하는 API입니다.

evas_object_polygon_add(Evas *) 는 캔버스에 Polygon 객체를 생성하는 API입니다.

evas_object_polygon_point_add(Evas_Object *, Evas_Coord, Evas_Coord) 는 Polygon 객체에 포인트 좌표를 추가하는 API입니다. Polygon은 최소한 3개 이상의 포인트를 가져야 합니다. 1번째 파라미터는 Polygon 객체, 2번째는 x축 좌표, 3번째는 y축 좌표입니다.

evas_object_color_set(Evas_Object *, int, int, int, int) 는 도형에 컬러를 지정하는 함수API입니다. 파라미터는 순서대로 도형 객체, Red 컬러, Green 컬러, Blue 컬러, 반투명도 입니다.

예제를 빌드하고 실행시켜 봅시다. 화면에 노란색 삼각형이 표시되었습니다.



2) 오각형 그리기

Polygon 객체에 4개의 포인트를 추가하면 사각형이 만들어지고, 5개의 포인트를 추가하면 오각형이 만들어집니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

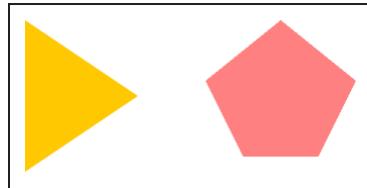
```
/* Polygon - Triangle */
Evas_Object *polygon = evas_object_polygon_add(canvas);
evas_object_polygon_point_add(polygon, 20, 50);
evas_object_polygon_point_add(polygon, 170, 150);
evas_object_polygon_point_add(polygon, 20, 250);
evas_object_color_set(polygon, 255, 200, 0, 255);
evas_object_show(polygon);

/* Polygon - Pentagon */
polygon = evas_object_polygon_add(canvas);
evas_object_polygon_point_add(polygon, 360, 50);
evas_object_polygon_point_add(polygon, 460, 130);
evas_object_polygon_point_add(polygon, 410, 230);
evas_object_polygon_point_add(polygon, 310, 230);
evas_object_polygon_point_add(polygon, 260, 130);
evas_object_color_set(polygon, 255, 128, 128, 255);
evas_object_show(polygon);

}
```

새로운 Polygon 객체를 생성해서 5개의 포인트를 추가하였습니다.

예제를 다시 실행시키면 화면에 분홍색 오각형이 추가되어 있습니다.



3) Polygon 으로 정다각형 그리기

정사각형 혹은 정육각형 처럼 변의 길이와 꼭지점의 각도가 모두 같은 도형을 정다각형이라고 합니다. 정다각형을 생성하는 함수를 만들어 봅시다. 수학함수를 사용하기 때문에 소스파일 위쪽에 라이브러리를 인클루드 합니다.

```
#include "drawpolygon.h"  
#include <math.h>
```

create_base_gui() 함수 위에 새로운 함수를 추가합니다. 정다각형을 생성하는 함수입니다.

```
static Evas_Object*  
create_circle(Evas* canvas, int x, int y, int radius, int r, int g, int b, int a, int edge_count)  
{  
    int x1, y1, x2, y2;  
    float angle=0.f;
```

```
Evas_Object *polygon = evas_object_polygon_add(canvas);
```

```

for(int i=0; i < edge_count; i++)
{
    angle = (M_PI * 2) / (float)edge_count * i;
    x1 = sin(angle) * radius + x;
    y1 = cos(angle) * radius + y;
    evas_object_polygon_point_add(polygon, x1, y1);
}
evas_object_color_set(polygon, r, g, b, a);
evas_object_show(polygon);
return polygon;
}

```

M_PI 는 Pi 값이 저장된 상수입니다.

sin(double) 은 사인값을 계산하는 API입니다. 각도 단위는 Pi입니다.
예를 들어 180도를 지정하려면 Pi를 전달하고, 90도를 지정하려면 Pi / 2를 전달하면 됩니다.

cos(double) 은 사인값을 계산하는 API입니다. 각도 단위는 Pi입니다.

위 함수를 사용해서 정팔각형을 만들어 보겠습니다. create_base_gui() 함수 끝부분에 새로운 코드 한줄을 추가합니다.

```

/* Polygon - Pentagon */
polygon = evas_object_polygon_add(canvas);
evas_object_polygon_point_add(polygon, 360, 50);
evas_object_polygon_point_add(polygon, 460, 130);
evas_object_polygon_point_add(polygon, 410, 230);
evas_object_polygon_point_add(polygon, 310, 230);
evas_object_polygon_point_add(polygon, 260, 130);
evas_object_color_set(polygon, 255, 128, 128, 255);

```

```

evas_object_show(polygon);

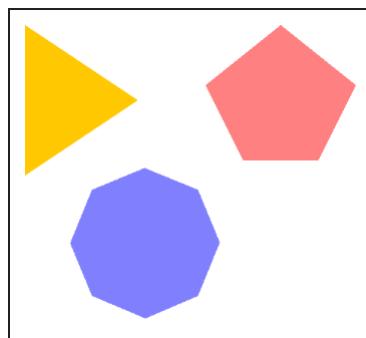
/* Polygon - 10 */
crate_circle(canvas, 180, 340, 100, 128, 128, 255, 255, 8);

}

```

crate_circle() 함수의 파라미터는 순서대로 Evas 객체, 중심점 x좌표, 중심점 y좌표, 반지름, Red 컬러, Green 컬러, Blue 컬러, 반투명 컬러, 포인트 갯수 입니다.

예제를 다시 실행시키면 보라색의 정팔각형이 추가되어 있습니다.



4) Polygon 으로 원 그리기

정다각형의 포인트 개수를 증가시키면 원 처럼 보이게 됩니다.
create_base_gui() 함수 끝부분에 새로운 코드 한줄을 추가합니다.

```

/* Polygon - 10 */
crate_circle(canvas, 180, 340, 100, 128, 128, 255, 255, 8);

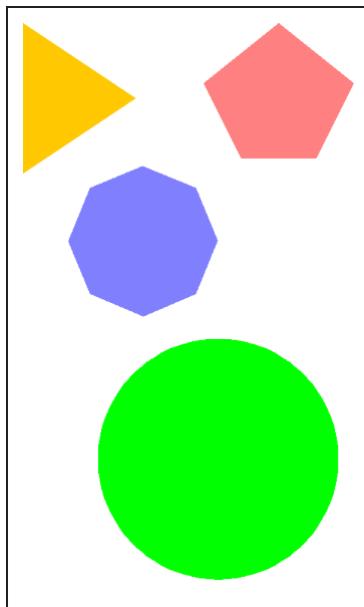
/* Polygon - Circle */
crate_circle(canvas, 280, 600, 160, 0, 255, 0, 255, 90);

}

```

Polygon에 90개의 포인트를 추가하였습니다. 결과가 어떻게 나오는지 확인해 보겠습니다.

예제를 다시 실행시키면 화면 아래쪽에 연두색 정90각형이 추가되었습니다. 포인트가 많아서 원 처럼 보입니다.



5) 관련 API

Evas *evas_object_evas_get(Evas_Object *obj) : Evas 객체를 생성하는 API.
/ 파라미터 - 윈도우 객체.

Evas_Object *evas_object_polygon_add(Evas *e) : 캔버스에
Polygon 객체를 생성하는 API.

void evas_object_polygon_point_add(Evas_Object *obj, Evas_Coord x,
Evas_Coord y) : Polygon 객체에 포인트 좌표를 추가하는 API.
/ 파라미터 : Polygon, x축 좌표, y축 좌표.

```
void evas_object_color_set(Evas_Object *obj, int r, int g, int b, int
```

a) : 도형에 컬러를 지정하는 API. / 파라미터 - 도형 객체, Red 컬러,
Green 컬러, Blue 컬러, 반투명도. 컬러값의 범위는 0~255 사이. 예를
들어서 노란색 컬러를 만들고 싶다면 255,255,0,255로 지정하면 됩니다.

34. 캔버스에 텍스트 출력

화면에 도형을 그릴 때는 캔버스를 사용하면 됩니다. Evas 는 EFL에서 제공하는 캔버스입니다. Evas에 그린 모든 도형은 객체로 생성됩니다. 이번 시간에는 캔버스에 텍스트 문자열을 출력하는 방법을 알아보겠습니다.

1) 캔버스에 텍스트 출력

새로운 소스 프로젝트를 생성하고 Project name을 CanvasTextColor 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수 위에 텍스트 객체를 생성하는 함수를 추가합니다.

```
// Create Text object
static Evas_Object *
create_text(Evas *canvas, Evas_Object *grid,
           Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h,
           const char *str, int font_size,
           int r, int g, int b, int a)
{
    Evas_Object *text = evas_object_text_add(canvas);
    evas_object_text_text_set(text, str);
    evas_object_text_font_set(text, "DejaVu", font_size);
    evas_object_color_set(text, r, g, b, a);
    elm_grid_pack(grid, text, x, y, w, h);
    evas_object_show(text);
}
```

`evas_object_text_add(Evas *)` 는 캔버스에 텍스트 객체를 생성하는 API입니다.

`evas_object_text_text_set(Evas_Object *, char *)` 는 텍스트 객체에 문자열을 지정하는 API입니다.

`evas_object_text_font_set(Evas_Object *, char *, Evas_Font_Size)` 는 텍스트 객체에 본트를 지정하는 API입니다. 1번째 파라미터는 텍스트 객체, 2번째는 폰트 종류, 3번째는 폰트 크기입니다.

방금 만든 함수를 사용해서 캔버스에 텍스트를 출력해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다. Label은 필요없으니 주석처리 합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);*/

/* child object - indent to how relationship */
/* A grid to stretch content within grid size */
```

```

Evas_Object *grid = elm_grid_add(ad->win);
elm_grid_size_set(grid, 480, 800);
evas_object_size_hint_weight_set(grid,
EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, grid);
evas_object_show(grid);

{ /* child object - indent to how relationship */
Evas* canvas = evas_object_evas_get(ad->win);

create_text(canvas, grid, 50, 100, 300, 100,
"Hello World!", 60, 80, 80, 255, 255);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

텍스트 객체의 영역좌표를 상대좌표로 지정하기 위해서 Grid 컨테이너를 생성하였습니다.

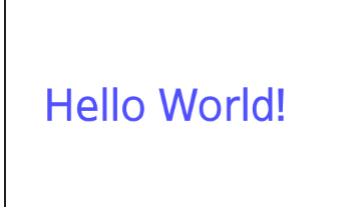
`elm_grid_size_set()` 은 Grid 사이즈를 변경하는 API 입니다. 기본 넓이는 100, 높이도 100 입니다. 위 코드에서는 480 x 800 으로 변경하였습니다. 기본 값 보다 더 세밀한 위치 설정이 가능합니다.

`evas_object_evas_get()` 함수를 사용해서 Evas 객체를 생성합니다.

`create_text()` 함수를 사용해서 캔버스에 텍스트 객체를 생성합니다. 파라미터는 순서대로 캔버스 객체, Grid, x좌표, y좌표, 넓이, 높이, 텍스트 문자열, 폰트 크기, Red 컬러, Green 컬러, Blue 컬러, 반투명 컬러입니다.

예제를 빌드하고 실행시켜 봅시다. 화면 보라색으로 'Hello World' 라는

텍스트가 출력되었습니다.



Hello World!

2) 텍스트에 그림자 적용

텍스트에 그림자 효과를 표현하려면 2개의 텍스트를 컬러와 위치를 달리 지정하면 됩니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다. 1번째 텍스트와 동일한 내용과 크기로 2번째 텍스트를 생성하였습니다.

```
{ /* child object - indent to how relationship */
    Evas* canvas = evas_object_evas_get(ad->win);

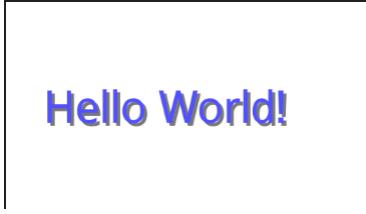
    create_text(canvas, grid, 54, 104, 300, 100,
                "Hello World!", 60, 120, 120, 120, 255);

    create_text(canvas, grid, 50, 100, 300, 100,
                "Hello World!", 60, 80, 80, 255, 255);

}
```

2번째 텍스트는 그림자에 해당하며 1번째 텍스트 보다 먼저 출력됩니다. 그리고 위치가 약간 다르며 컬러도 회색입니다.

예제를 다시 실행시켜 봅시다. 텍스트에 그림자 효과가 적용되었습니다.



Hello World!

3) 관련 API

Evas *evas_object_evas_get(Evas_Object *obj) : Evas 객체를 생성하는 API.
/ 파라미터 - 윈도우 객체.

Evas_Object *evas_object_text_add(Evas *e) : 캔버스에 텍스트 객체를
생성하는 API.

void evas_object_text_text_set(Evas_Object *obj, const char
*text) : 텍스트 객체에 문자열을 지정하는 API.

void evas_object_text_font_set(Evas_Object *obj, const char *font,
Evas_Font_Size size) : 텍스트 객체에 폰트를 지정하는 API. 파라미터 :
텍스트 객체, 폰트 종류, 폰트 크기.

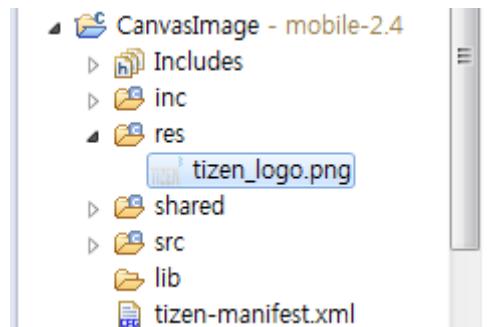
void evas_object_color_set(Evas_Object *obj, int r, int g, int b, int
a) : 도형에 컬러를 지정하는 API. / 파라미터 - 도형 객체, Red 컬러,
Green 컬러, Blue 컬러, 반투명도. 컬러값의 범위는 0~255 사이. 예를
들어서 노란색 컬러를 만들고 싶다면 255,255,0,255로 지정하면 됩니다.

35. 캔버스에 이미지 출력

화면에 도형을 그릴 때는 캔버스를 사용하면 됩니다. Evas 는 EFL에서 제공하는 캔버스입니다. Evas에 그린 모든 도형은 객체로 생성됩니다. 이번 시간에는 캔버스에 이미지를 출력하는 방법을 알아보겠습니다.

1) 캔버스에 이미지 출력

새로운 소스 프로젝트를 생성하고 Project name을 CanvasImage 로 지정합니다. 이번 예제에서는 이미지 파일이 필요합니다. 부록 /Image 폴더에 있는 tizen_logo.png 파일을 소스 프로젝트 /res 폴더로 복사합니다.



방금 복사한 이미지 파일을 사용하려면 파일의 절대 경로를 구해야 합니다. src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수 위에 새로운 함수를 추가합니다. /res 폴더에 저장된 파일의 절대 경로를 반환하는 함수입니다.

```
static void  
app_get_resource(const char *res_file_in, char *res_path_out, int res_path_max)  
{
```

```

char *res_path = app_get_resource_path();
if (res_path) {
    snprintf(res_path_out, res_path_max, "%s%s", res_path, res_file_in);
    free(res_path);
}

```

create_base_gui() 함수로 이동해서 새로운 코드를 추가합니다. 캔버스를 생성하고 이미지 파일을 로딩해서 IMAGE 객체를 생성하는 코드입니다. Label 생성 코드는 주석처리 합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);*/

{ /* child object - indent to how relationship */
/* A grid to stretch content within grid size */
Evas_Object *grid = elm_grid_add(ad->win);
elm_grid_size_set(grid, 480, 800);
evas_object_size_hint_weight_set(grid, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
}
```

```

EVAS_HINT_EXPAND);

    elm_object_content_set(ad->conform, grid);
    evas_object_show(grid);

}

/* Canvas */
Evas* canvas = evas_object_evas_get(ad->win);

char img_path[PATH_MAX] = "";
app_get_resource("tizen_logo.png", img_path, PATH_MAX);

/* Image-1 */
Evas_Object *img = evas_object_image_filled_add(canvas);
evas_object_image_file_set(img, img_path, NULL);
elm_grid_pack(grid, img, 40, 10, 400, 280);
evas_object_show(img);

}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

이미지 객체를 상대좌표로 지정하기 위해서 Grid 컨테이너를 생성했습니다. 세밀하게 영역좌표를 지정하기 위해서 elm_grid_size_set() 함수를 사용해서 Grid 사이즈를 480x800 으로 지정해습니다.

evas_object_image_filled_add(Evas *) 는 원본 이미지를 빈영역 없이 채우는 IMAGE 객체를 생성하는 API 입니다.

evas_object_image_file_set(Evas_Object *, char *, char *) 는 IMAGE 객체에 이미지 파일을 로딩하는 API 입니다.

예제를 빌드하고 실행시켜 봅시다. 화면에 이미지가 출력됩니다. 원본

이미지는 수평으로 길지만 위아래로 늘어났습니다.



2) 캔버스에 타일 형식으로 이미지 출력

이번에는 주어진 영역에 이미지를 타일 형식으로 배치하는 방법을 알아보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다. 2번째 IMAGE 객체를 생성하는 코드입니다.

```
/* Image-1 */
Evas_Object *img = evas_object_image_filled_add(canvas);
evas_object_image_file_set(img, img_path, NULL);
elm_grid_pack(grid, img, 40, 10, 400, 280);
evas_object_show(img);

/* Image-2 */
int w, h;
img = evas_object_image_add(canvas);
evas_object_image_file_set(img, img_path, NULL);
evas_object_image_size_get(img, &w, &h);
evas_object_image_fill_set(img, 110, 37, w, h);
elm_grid_pack(grid, img, 40, 310, 400, 280);
evas_object_show(img);

}
```

`evas_object_image_add(Evas *)` 는 이미지를 타일 형식으로 출력하는 IMAGE 객체를 생성하는 API 입니다. 이미지의 크기와 시작 위치를 지정해 주어야 합니다.

`evas_object_image_size_get(const Evas_Object *, int *, int *)` 는 원본 이미지의 크기를 반환하는 API 입니다. 1번째 파라미터는 IMAGE 객체, 2번째에는 원본 이미지의 넓이를 반환하고, 3번째에는 원본 이미지의 높이를 반환합니다.

`evas_object_image_fill_set(Evas_Object *, Evas_Coord, Evas_Coord, Evas_Coord, Evas_Coord)` 는 IMAGE 객체에 원본 이미지의 출력 위치와 크기를 지정하는 API 입니다. 파라미터는 순서대로 IMAGE 객체, x좌표, y좌표, 넓이, 높이 입니다.

예제를 다시 실행시켜 봅시다. 타일 형식으로 이미지가 배치됩니다.



3) 원본 비율 대로 최대한 크게 이미지 출력

지정된 영역에 이미지를 최대한 크게 출력해 봅시다. 원본 이미지의 가로/세로 비율을 그대로 유지한 상태로 출력하려면 계산을 해주어야 합니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다. 출력 이미지의 크기를 계산해서 IMAGE 객체를 생성하는 함수입니다.

```
// Create IMAGE object. Source image's rate is no change
static Evas_Object *
create_image(Evas *canvas, Evas_Object *grid, const char *img_path, int x, int y, int w, int
h)
{
    int source_w, source_h, new_x, new_y, new_w, new_h;
    float rate_h, rate_v, rate;

    // Create IMAGE object
    Evas_Object *img = evas_object_image_add(canvas);
    // Set source image file
    evas_object_image_file_set(img, img_path, NULL);
    // Get source image size
    evas_object_image_size_get(img, &source_w, &source_h);
    // Load failed - zero sized image
    if ((source_w == 0) || (source_h == 0))
    {
        evas_object_del(img);
        return NULL;
    }

    // Calculage Zoom rate
    rate_h = (float)w / (float)source_w;
    rate_v = (float)h / (float)source_h;
    rate = (rate_h < rate_v) ? rate_h : rate_v;
```

```

// Calculate output image size
new_w = source_w * rate;
new_h = source_h * rate;
evas_object_image_fill_set(img, 0, 0, new_w, new_h);

// Calculate output Image position
new_x = x + (w - new_w) / 2;
new_y = y + (h - new_h) / 2;
elm_grid_pack(grid, img, new_x, new_y, new_w, new_h);

evas_object_show(img);
return img;
}

```

방금 만든 함수를 호출해서 3번째 IMAGE 객체를 생성해 봅시다.
create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```

/* Image-2 */
int w, h;
img = evas_object_image_add(canvas);
evas_object_image_file_set(img, img_path, NULL);
evas_object_image_size_get(img, &w, &h);
evas_object_image_fill_set(img, 110, 37, w, h);
elm_grid_pack(grid, img, 40, 310, 400, 280);
evas_object_show(img);

/* Image-3 */
create_image(canvas, grid, img_path, 40, 610, 400, 180);
}

```

`create_image()` 함수의 파라미터는 캔버스 객체, 이미지 파일 경로, x좌표, y좌표, 넓이, 높이 순입니다.

예제를 다시 실행시켜 봅시다. 화면 아래쪽에 3번째 이미지가 출력되었고 원본 이미지의 가로/세로 비율이 그대로 유지되었습니다.



4) 관련 API

`Evas *evas_object_evas_get(Evas_Object *obj)` : Evas 객체를 생성하는 API.
/ 파라미터 - 윈도우 객체.

`Evas_Object *evas_object_image_filled_add(Evas *e)` : 원본 이미지를
빈영역 없이 채우는 IMAGE 객체를 생성하는 API.

`void evas_object_image_file_set(Evas_Object *obj, const char *file, const
char *key)` : IMAGE 객체에 이미지 파일을 로딩하는 API.

`Evas_Object *evas_object_image_add(Evas *e)` : 이미지를 파일 형식으로
출력하는 IMAGE 객체를 생성하는 API. 이미지의 크기와 시작 위치를

지정해 주어야 합니다.

void evas_object_image_size_get(const Evas_Object *obj, int *w, int *h) : 원본 이미지의 크기를 반환하는 API. 1번째 파라미터는 IMAGE 객체, 2번째에는 원본 이미지의 넓이를 반환하고, 3번째에는 원본 이미지의 높이를 반환합니다.

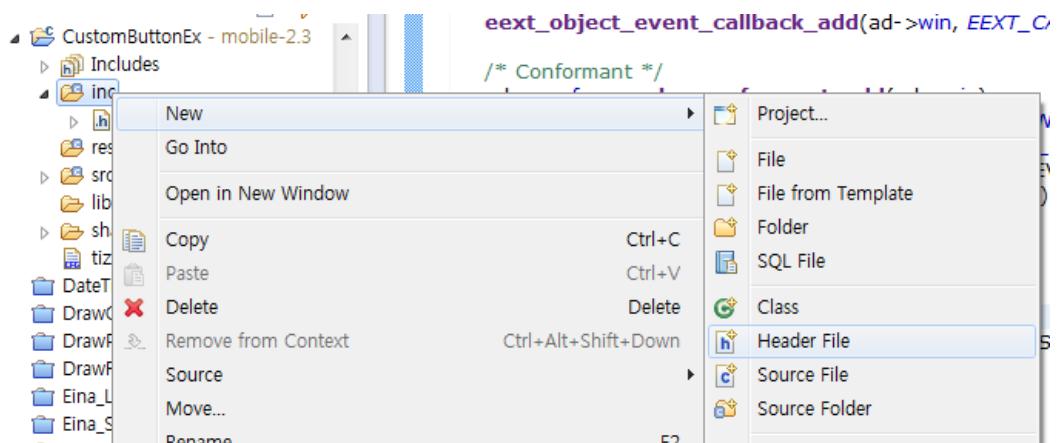
void evas_object_image_fill_set(Evas_Object *obj, Evas_Coord x, Evas_Coord y, Evas_Coord w, Evas_Coord h) : IMAGE 객체에 원본 이미지의 출력 위치와 크기를 지정하는 API. / 파라미터 : IMAGE 객체, x좌표, y좌표, 넓이, 높이.

36. 커스텀 Button 제작

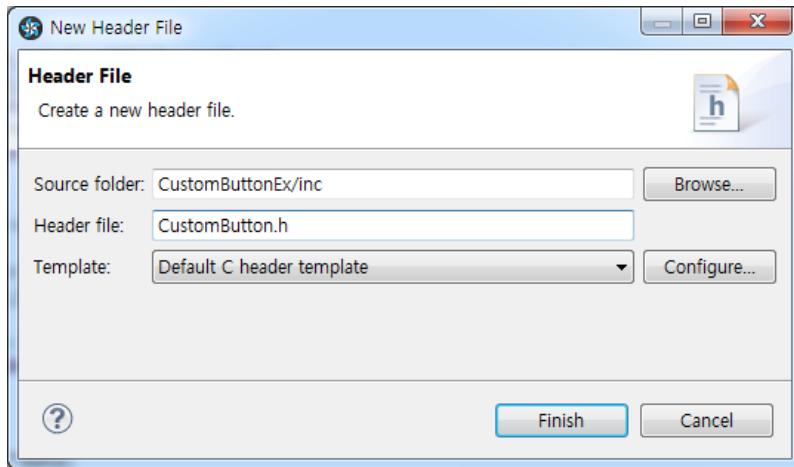
사용자들의 다양한 요구사항을 만족시키려면 시스템에서 제공하는 기본 위젯 만으로는 부족한 경우가 많습니다. 그런 경우에는 직접 커스텀 위젯을 제작해야 합니다. 이번 시간에는 커스텀 Button 위젯을 직접 만들어 보겠습니다.

1) 캔버스에 텍스트 출력

새로운 소스 프로젝트를 생성하고 Project name을 CustomButtonEx으로 지정합니다. 소스 프로젝트가 생성되었으면 커스텀 Button 라이브러리 파일을 생성하겠습니다. /inc 폴더를 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [New > Header File]을 선택합니다.



팝업창이 나타나면 Header file에 CustomButton.h 이라고 입력하고 Finish 버튼을 누릅니다.



커스텀 위젯에 배경 사각형을 출력하는 기능을 구현해 보겠습니다.
/inc/CustomButton.h 파일이 생성되면 소스코드를 추가합니다.
라이브러리 헤더파일을 선언하고, 데이터 구조체를 정의하고, 배경
사각형을 생성하는 코드입니다.

```
#ifndef CUSTOMBUTTON_H_
#define CUSTOMBUTTON_H_

#include <app.h>
#include <Elementary.h>
#include <system_settings.h>
#include <efl_extension.h>
#include <dlog.h>

typedef struct buttondata {
    Evas_Object *rect;
    Evas_Object *text;
} buttondata_s;

Evas_Object*
```

```

create_rect(Evas* canvas, Evas_Object* grid, int x, int y, int w, int h,
           int r, int g, int b, int a)
{
    Evas_Object *rect = evas_object_rectangle_add(canvas);
    evas_object_color_set(rect, r, g, b, a);
    elm_grid_pack(grid, rect, x, y, w, h);
    evas_object_show(rect);
    return rect;
}

buttondata_s*
create_button(Evas* canvas, Evas_Object* grid, Evas_Coord x, Evas_Coord y,
              Evas_Coord w, Evas_Coord h,
              const char* str, Evas_Object_Event_Cb func)
{
    buttondata_s* bd = (buttondata_s*)malloc( sizeof( buttondata_s ) );

    /* Rectangle */
    bd->rect = create_rect(canvas, grid, x, y, w, h, 128, 128, 255, 255);

    return bd;
}

#endif /* CUSTOMBUTTON_H_ */

```

buttondata는 커스텀 위젯에서 사용하는 데이터 구조체입니다. rect는 위젯의 영역좌표입니다. text에는 캡션 텍스트 문자열을 저장합니다.

create_rect() 는 캔버스에 Rect 객체를 생성하는 함수입니다. 파라미터는 순서대로 캔버스 객체, x좌표, y좌표, 넓이, 높이, Red 컬러, Green 컬러, Blue 컬러, 반투명 컬러 입니다.

evas_object_rectangle_add(Evas *) 는 캔버스에 Rectangle 객체를 생성하는 API입니다.

`evas_object_color_set(Evas_Object *, int, int, int, int)` 는 도형에 컬러를 지정하는 API입니다. 파라미터는 순서대로 Red, Green, Blue, 반투명도입니다. 255, 0, 0, 192 로 지정하면 반투명이 적용된 빨간색 컬러가 만들어 집니다.

`create_button()` 는 커스텀 Button 위젯을 생성하는 함수입니다. 외부에서 이 함수를 호출하면 됩니다. 파라미터는 순서대로 캔버스, x좌표, y좌표, 넓이, 높이, 캡션 텍스트, 콜백 이벤트 함수명 입니다.

`malloc(size_t)` 는 주어진 사이즈 만큼 메모리를 확보해서 반환하는 API입니다.

`sizeof()` 는 특정 객체 혹은 구조체의 메모리 용량을 구해서 반환하는 API입니다.

Main 화면에서 커스텀 위젯 객체를 생성해 봅시다. src 폴더에 소스파일(~.c)을 열고 새로운 코드를 추가합니다.

```
#include "custombuttonex.h"
#include "custombutton.h"

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
} appdata_s;
```

```
buttondata_s* m_bd1;
```

커스텀 Button 위젯 파일을 include 하고, buttondata 구조체를 전역변수로 생성합니다.

그런 다음 create_base_gui() 함수로 이동해서 새로운 코드를 추가합니다. 캔버스 객체를 생성하고 커스텀 Button 위젯을 생성하는 코드입니다. Label 생성코드는 주석 처리합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);*/

{ /* child object - indent to how relationship */
/* A grid to stretch content within grid size */
Evas_Object *grid = elm_grid_add(ad->win);
elm_grid_size_set(grid, 480, 800);
evas_object_size_hint_weight_set(grid, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, grid);
evas_object_show(grid);

{
```

```

/* Canvas */
Evas* canvas = evas_object_evas_get(ad->conform);

/* Custom Button-1 */
m_bd1 = create_button(canvas, grid, 50, 200, 300, 100, "Button-1",
NULL);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

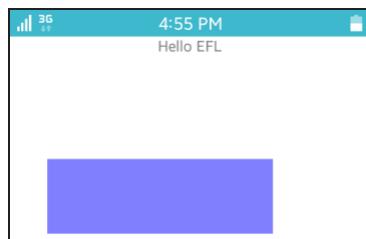
```

오브젝트를 상대좌표로 지정하기 위해서 Grid 컨테이너를 생성하였습니다. elm_grid_size_set() 함수를 사용해서 Grid 사이즈를 480x800 으로 변경하였습니다.

evas_object_evas_get(Evas_Object *) 는 Evas 객체를 생성하는 API입니다.

create_button() 은 좀전에 커스텀 Button 위젯 파일(CustomButton.h)에서 만들었던 커스텀 Button 생성 함수입니다.

예제를 빌드하고 실행시켜 봅시다. 화면에 하늘색 사각형이 표시되었습니다. 커스텀 Button의 배경 사각형 입니다.



2) 캡션 텍스트 출력

사각형 배경 위에 캡션 텍스트를 출력해 보겠습니다. 텍스트 출력에는 TextBlock을 사용해 보겠습니다. TextBlock 은 html 태그로 폰트 크기, 컬러 같은 텍스트 속성을 지정할 수 있는 캔버스 객체입니다. CustomButton.h 파일로 되돌아 가서 create_button() 함수 위에 새로운 함수를 추가합니다.

```
// Create TextBlock object
Evas_Object*
create_textblock(Evas* canvas, Evas_Object* grid, Evas_Coord x, Evas_Coord y, Evas_Coord
w, Evas_Coord h, const char* str)
{
    Evas_Object *textblock = evas_object_textblock_add(canvas);
    elm_grid_pack(grid, textblock, x, y, w, h);

    Evas_Textblock_Style *st = evas_textblock_style_new();
    evas_textblock_style_set(st, "DEFAULT='font=Sans font_size=50 color=#eee
wrap=mixed align=center'");
    evas_object_textblock_style_set(textblock, st);
    evas_textblock_style_free(st);
    evas_object_textblock_text_markup_set(textblock, str);
    evas_object_show(textblock);

    return textblock;
}
```

evas_object_textblock_add(Evas *) 는 TextBlock 객체를 생성하는 API입니다.

Evas_Textblock_Style 는 TextBlock의 속성 정보를 저장하는 스타일 구조체입니다.

evas_textblock_style_new() 는 Evas_Textblock_Style 객체를 생성하는 API입니다.

evas_textblock_style_set(Evas_Textblock_Style *, char *) 는 Evas_Textblock_Style에 스타일을 정의하는 API입니다. 1번째 파라미터에는 Evas_Textblock_Style 객체를 전달하고, 2번째 파라미터에는 html 태그로 텍스트 속성을 지정하면 됩니다.

font=Sans 는 산세리프, 고딕체를 지정하는 태그입니다.

font_size=50 는 폰트 크기를 50으로 지정하는 태그입니다.

color=#eee 는 폰트 크기를 회색으로 지정하는 태그입니다.

align=center 는 수평 정렬방식을 가운데로 지정하는 태그입니다.

evas_object_textblock_style_set(Evas_Object *, Evas_Textblock_Style *) 는 TextBlock 객체에 스타일 객체를 지정하는 API입니다. 1번째 파라미터에는 TextBlock 객체를 전달하고, 2번째에는 Evas_Textblock_Style 객체를 전달합니다.

evas_textblock_style_free(Evas_Textblock_Style *) 는 Evas_Textblock_Style 객체를 삭제하는 API입니다.

evas_object_textblock_text_markup_set(Evas_Object *, const char *) 는 TextBlock 객체에 텍스트 문자열을 지정하는 함수API입니다.

이 함수를 create_textblock() 함수에서 호출해 주면 됩니다.
create_textblock() 함수에 새로운 코드를 추가합니다.

```

bd->rect = create_rect(canvas, grid, x, y, w, h, 128, 128, 255);

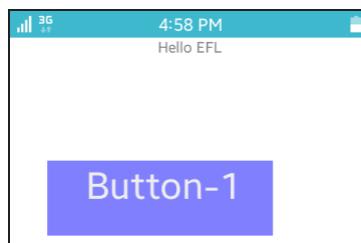
/* Text */
bd->text = create_textblock(canvas, grid, x, y, w, h, str);

return bd;

```

`create_textblock()` 함수의 파라미터는 순서대로 캔버스 객체, x좌표, y좌표, 높이 넓이, 텍스트 문자열입니다.

예제를 다시 실행시켜 봅시다. 사각형 영역에 캡션 텍스트가 표시되었습니다.



3) 캡션 텍스트 중앙 정렬

캡션 텍스트가 사각형 위쪽에 표시되어 있습니다. 중앙으로 위치를 변경해 보겠습니다. `create_textblock()` 함수에 새로운 코드를 추가합니다.

```

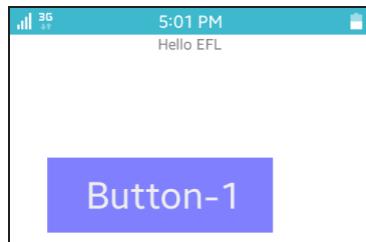
Evas_Textblock_Style *st = evas_textblock_style_new();
evas_textblock_style_set(st,    "DEFAULT='font=Sans    font_size=50    color=#eee
wrap=mixed align=center'");
evas_object_textblock_style_set(textblock, st);
evas_textblock_style_free(st);

```

```
evas_object_textblock_valign_set(textblock, 0.5);
evas_object_textblock_text_markup_set(textblock, str);
evas_object_show(textblock);
```

evas_object_textblock_valign_set(Evas_Object *, double) 는 TextBlock 캡션 텍스트의 수직 정렬방식을 지정하는 API입니다. 2번째 파라미터에 0을 전달하면 위쪽 정렬, 1을 전달하면 아래쪽 정렬, 0.5를 전달하면 중앙정렬이 됩니다.

예제를 다시 실행시켜 봅시다. 이번에는 캡션 텍스트가 중앙에 표시되었습니다.



4) Button 클릭 이벤트 구하기

커스텀 Button을 클릭하면 배경 컬러가 변경되는 기능을 구현해 보겠습니다. Touch 이벤트를 구하면 됩니다. `create_button()` 함수 끝부분에 새로운 코드를 추가합니다.

```
bd->text = create_textblock(canvas, grid, x, y, w, h, str);

evas_object_event_callback_add(    bd->text,    EVAS_CALLBACK_MOUSE_DOWN,
on_mouse_down, (void*)bd);
evas_object_event_callback_add(    bd->text,    EVAS_CALLBACK_MOUSE_UP,
```

```
on_mouse_up, (void*)bd);
```

```
    return bd;
```

TextBlock에 Touch 다운 이벤트가 발생하면 on_mouse_down 함수를 호출합니다.

TextBlock에 Touch 해제 이벤트가 발생하면 on_mouse_up 함수를 호출합니다.

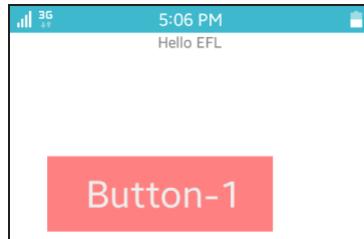
이제 콜백 함수를 만들 차례입니다. create_button() 함수 위에 새로운 함수 2개를 추가합니다.

```
// Touch Down event callback
void
on_mouse_down(void *data, Evas *e, Evas_Object *button, void *event_info)
{
    buttondata_s* bd = (buttondata_s*)data;
    evas_object_color_set(bd->rect, 255, 128, 128, 255);
}
```

```
// Touch Up event callback
void
on_mouse_up(void *data, Evas *e, Evas_Object *button, void *event_info)
{
    buttondata_s* bd = (buttondata_s*)data;
    evas_object_color_set(bd->rect, 128, 128, 255, 255);
}
```

사용자가 커스텀 Button을 Touch 다운하면 배경 컬러를 분홍색으로 변경하고, Touch 해제하면 배경 컬러를 다시 하늘색으로 변경하는 코드입니다.

예제를 다시 실행시켜 봅시다. Button을 클릭했다가 해제하면 배경 컬러가 변경됩니다.



5) Main 화면에서 클릭 이벤트 구하기

Main 화면에서 Button 클릭 이벤트를 구하려면 Touch 해제 이벤트가 발생할 때 콜백 함수를 호출해야 합니다. `create_button()` 함수 끝부분에 새로운 코드를 추가합니다.

```
    evas_object_event_callback_add(      bd->text,      EVAS_CALLBACK_MOUSE_DOWN,  
on_mouse_down, (void*)bd);  
  
    evas_object_event_callback_add( bd->text, EVAS_CALLBACK_MOUSE_UP, on_mouse_up,  
(void*)bd);  
  
    evas_object_event_callback_add( bd->text, EVAS_CALLBACK_MOUSE_UP, func,  
(void*)bd);  
  
    return bd;
```

사용자가 Button을 클릭했다가 해제하면 Main 화면에서 전달한 함수가

호출됩니다.

Main 소스파일(custombuttonex.c)로 이동해서 create_base_gui() 함수에 코드를 수정합니다.

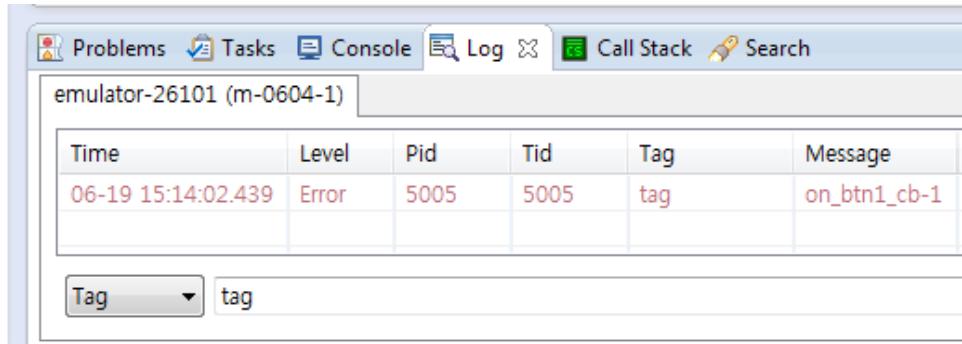
```
{  
    /* Canvas */  
    Evas* canvas = evas_object_evas_get(ad->conform);  
  
    /* Custom Button-1 */  
    //m_bd1 = create_button(canvas, 50, 200, 300, 100, "Button-1", NULL);  
    m_bd1 = create_button(canvas, grid, 50, 200, 300, 100, "Button-1",  
on_btn1_cb);  
}
```

커스텀 Button의 콜백 함수명을 on_btn1_cb 로 지정하였습니다. 이제 콜백 함수를 추가할 차례입니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void  
on_btn1_cb(void *data, Evas *e, Evas_Object *button, void *event_info)  
{  
    dlog_print(DLOG_ERROR, "tag", "on_btn1_cb-1");  
}
```

사용자가 커스텀 Button을 클릭하면 Log 메시지를 출력하는 코드입니다.

예제를 다시 실행시켜 봅시다. Button을 클릭하면 Log 판넬에 on_btn1_cb-1 이라는 메시지가 출력됩니다.



6) 캡션 텍스트 변경하기

2번째 커스텀 Button을 추가하고 클릭하면 1번째 Button의 캡션 텍스트가 변경되는 기능을 구현해 보겠습니다. 그러기 위해서 커스텀 위젯 파일(CustomButton.h)에 캡션 텍스트 변경 함수를 추가해야 합니다. 위치는 상관없지만 이렇게 외부에서 호출하는 함수는 가능한 아래쪽에 배치하는 것이 좋습니다. 이 함수에서 다른 함수를 호출할 수도 있기 때문입니다.

```
void set_button_text(buttondata_s* bd, const char* str)
{
    evas_object_textblock_text_markup_set(bd->text, str);
}

#endif /* CUSTOMBUTTON_H_ */
```

Main 화면에서 이 함수를 호출하면 해당 Button의 캡션 텍스트가 변경됩니다. Main 소스파일(custombuttonex.c)로 이동해서 create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
{  
    /* Canvas */  
    Evas* canvas = evas_object_evas_get(ad->conform);  
  
    /* Custom Button-1 */  
    m_bd1 = create_button(canvas, grid, 50, 200, 300, 100, "Button-1", on_btn1_cb);  
  
    /* Custom Button-2 */  
    create_button(canvas, grid, 50, 400, 300, 100, "Button-2", on_btn2_cb);  
}
```

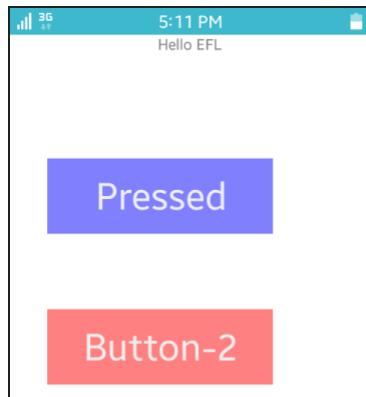
2번째 커스텀 Button을 생성하고 캡션 텍스트를 "Button-2", 콜백 함수를 on_btn2_cb로 지정하는 코드입니다.

이제 마지막으로 2번째 Button의 콜백 함수를 추가하면 됩니다.
create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static void  
on_btn2_cb(void *data, Evas *e, Evas_Object *button, void *event_info)  
{  
    set_button_text(m_bd1, "Pressed");  
}
```

사용자가 2번째 Button을 클릭하면 1번째 Button의 캡션 텍스트를 "Pressed"로 변경하는 코드입니다.

예제를 다시 실행시켜 봅시다. 2번째 Button을 클릭하면 1번째 Button의 캡션 텍스트가 변경됩니다.



7) 관련 API

Evas_Object *evas_object_textblock_add(Evas *e) : TextBlock 객체를 생성하는 API.

Evas_Textblock_Style : TextBlock의 속성 정보를 저장하는 스타일 구조체.

Evas_Textblock_Style *evas_textblock_style_new() : Evas_Textblock_Style 객체를 생성하는 API.

void evas_textblock_style_set(Evas_Textblock_Style *ts, const char *text) : Evas_Textblock_Style에 스타일을 정의하는 API. / 파라미터 : Evas_Textblock_Style 객체, html 태그.

void evas_object_textblock_style_set(Evas_Object *obj, Evas_Textblock_Style *ts) : TextBlock 객체에 스타일 객체를 지정하는 API. / 파라미터 : TextBlock 객체, Evas_Textblock_Style 객체.

void evas_textblock_style_free(Evas_Textblock_Style *ts) : Evas_Textblock_Style 객체를 삭제하는 API.

`void evas_object_textblock_text_markup_set(Evas_Object *obj, const char *text)` : TextBlock 객체에 텍스트 문자열을 지정하는 API.

`void evas_object_textblock_valign_set(Evas_Object *obj, double align)` : TextBlock 캡션 텍스트의 수직 정렬방식을 지정하는 API. 2번째 파라미터에 0을 전달하면 위쪽 정렬, 1을 전달하면 아래쪽 정렬, 0.5를 전달하면 중앙정렬이 됩니다.

37. Animator 사용방법

Animator를 사용하면 일정 시간 간격 마다 화면상의 객체에 변화를 줄수 있습니다. Timer와 비슷하지만 여러 가지 효과를 줄 수 있다는 점에서 Timer와 차이점이 있습니다. 예제를 통해서 구체적인 사용방법을 알아보겠습니다.

1) 이동 애니메이션

새로운 소스 프로젝트를 생성하고 Project name을 AnimatorEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 소스파일 위쪽에 변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
Evas_Object *rect1;
Evas_Object *rect2;
} appdata_s;

Eina_Bool anim_continue = ECORE_CALLBACK_RENEW;
Ecore_Pos_Map pos_map = ECORE_POS_MAP_LINEAR;
```

rect1과 rect2는 애니메이션을 적용할 사각형 객체입니다.

anim_continue는 애니메이션을 계속할지 여부를 결정하는 Boolean 형식의 변수입니다.

Ecore_Pos_Map은 애니메이션 스타일을 지정하는 옵션 타입입니다.
TimeLine 애니메이션에서 사용해 보겠습니다.

1번째 사각형 객체와 Animator 객체를 생성하겠습니다. create_base_gui() 함수에 새로운 코드를 추가합니다. Conformant와 Label 생성 코드는 주석처리 합니다.

```
/* Conformant */
/*ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);*/

/* Label*/
/*ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "Hello EFL");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);
evas_object_show(ad->label);*/

/* Evas */
Evas *evas = evas_object_evas_get(ad->win);

/* Rect-1 */
ad->rect1 = evas_object_rectangle_add(evas);
evas_object_pass_events_set(ad->rect1, EINA_TRUE);
evas_object_color_set(ad->rect1, 0, 0, 160, 160);
evas_object_resize(ad->rect1, 50, 50);
evas_object_show(ad->rect1);
```

```
/* Animation-1 */
Ecore_Animator *anim = ecore_animator_add(on_next_frame1, ad->rect1);
ecore_animator_frametime_set(1. / 60);

/* Show window after base gui is set up */
evas_object_show(ad->win);
```

Rectangle 객체를 생성하기 위해서 Evas 객체를 생성합니다.

ecore_animator_add(Ecore_Task_Cb, void *) 는 Animator 객체를 생성하는 API입니다. 1번째 파라미터는 프레임 이벤트 콜백 함수이고 2번째 파라미터는 사용자 데이터입니다. 주로 애니메이션을 적용할 객체 혹은 appdata를 전달합니다.

ecore_animator_frametime_set(double) 는 애니메이션 프레임 시간 간격을 지정하는 API입니다. 단위는 초단위이며, 1/60을 지정하면 1초에 60번 프레임 이벤트가 발생합니다.

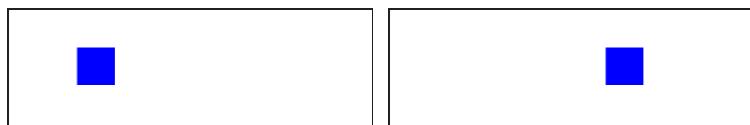
프레임 이벤트 함수를 만들 차례입니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static Eina_Bool on_next_frame1(void *data)
{
    static int x = 0;
    if (x >= 350)
        x = 0;
    evas_object_move(data, x += 2, 50);
    return anim_continue;
}
```

`evas_object_move()` 는 객체의 위치를 변경하는 API입니다. 매 프레임마다 x좌표를 2만큼 증가시킵니다. 그리고 x좌표가 350을 넘어서면 0에서 다시 시작합니다.

프레임 이벤트 함수에서 반환하는 값이 `ECORE_CALLBACK_RENEW` 이면 애니메이션을 계속하고, `ECORE_CALLBACK_CANCEL`을 반환하면 애니메이션이 중지됩니다.

예제를 빌드하고 실행시켜 봅시다. 파란색 사각형이 왼쪽에서 오른쪽으로 이동합니다. 어느 정도 이동하고 나면 다시 왼쪽 끝에서 이동을 계속합니다.



2) 애니메이션 중지

애니메이션을 중지하려면 프레임 이벤트 함수에서 `ECORE_CALLBACK_CANCEL`을 반환하면 됩니다. Button을 누르면 애니메이션을 중지하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 생성합니다. Table 컨테이너에 위젯을 추가하는 함수입니다.

```
static void
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)
{
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, 0.5);
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
    elm_table_pack(table, child, x, y, w, h);
```

```
    evas_object_show(child);
}


```

그런 다음 `create_base_gui()` 함수에 새로운 코드를 추가합니다. Box, Table을 생성하고 Button을 추가하는 코드입니다.

```
/* Rect-1 */
ad->rect1 = evas_object_rectangle_add(evas);
evas_object_pass_events_set(ad->rect1, EINA_TRUE);
evas_object_color_set(ad->rect1, 0, 0, 160, 160);
evas_object_resize(ad->rect1, 50, 50);
evas_object_show(ad->rect1);

{

/* Box to put the table in so we can bottom-align the table
 * window will stretch all resize object content to win size */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, box);
evas_object_show(box);

/* Table */
Evas_Object *table = elm_table_add(ad->win);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 20 * elm_config_scale_get(), 10 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fiill width but align to bottom of box */


```

```

evas_object_size_hint_align_set(table, EVAS_HINT_FILL, 1.0);
elm_box_pack_end(box, table);
evas_object_show(table);

{

/* Button-1 */
Evas_Object *btn = elm_button_add(ad->win);
elm_object_text_set(btn, "■");
evas_object_smart_callback_add(btn, "clicked", btn_stop_cb, NULL);
my_table_pack(table, btn, 0, 0, 2, 1);

}

}

evas_object_raise(ad->rect1);

/* Animation-1 */
Ecore_Animator *anim = ecore_animator_add(on_next_frame1, ad->rect1);

```

evas_object_raise() 는 특정 오브젝트를 최상위로 이동시켜서 다른 오브젝트에 가려지지 않도록 하는 API입니다.

Button을 눌렀을 때 전역 변수 anim_continue에 ECORE_CALLBACK_CANCEL를 저장하면 됩니다. create_base_gui() 함수 위에 Button 콜백 함수를 만들어 보겠습니다.

```

static void
btn_stop_cb(void *data, Evas_Object *obj, void *event_info)
{
    anim_continue = ECORE_CALLBACK_CANCEL;
}

```

Button을 누르면 on_next_frame1() 함수에서
ECORE_CALLBACK_CANCEL를 반환하게 되고 결국 애니메이션은
중지되는 것입니다.

예제를 다시 실행시켜 봅시다. Button을 누르면 사각형이 멈춥니다.



3) Timer로 애니메이션 일시 정지 & 재시작

애니메이션을 일시 정지하려면 ecore_animator_freeze() 함수를 사용하면 되고, 재시작하려면 ecore_animator_thaw() 함수를 사용하면 됩니다.
create_base_gui() 함수에 새로운 코드를 추가합니다. 2개의 Timer를 생성하는 코드입니다.

```
/* Animation-1 */
Ecore_Animator *anim = ecore_animator_add(on_next_frame1, ad->rect1);
/* add 2 timers to go off every 6 seconds */
ecore_timer_add(6, freeze_anim, anim);
Ecore_Timer *timer = ecore_timer_add(6, thaw_anim, anim);
/* delay the last timer by 3 seconds so the 2 timers are offset */
ecore_timer_delay(timer, 3);
```

2개의 Timer를 생성했습니다. 1번째 타이머는 일시정지를 담당하고 2번째 타이머는 재시작을 담당합니다.

ecore_timer_delay() 는 일정 시간이 지난후에 타이머를 시작하는 API입니다. 결과적으로 2번째 타이머는 $3 + 6 = 9$ 초 후에 이벤트가 발생하고 그 후에는 6초마다 이벤트가 발생합니다.

이제 타이머 이벤트 함수를 생성할 차례입니다. create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

```
static Eina_Bool freeze_anim(void *data)
{
    ecore_animator_freeze(data);
    // Animation stop Timer delete
    return ECORE_CALLBACK_CANCEL;
}
```

```
static Eina_Bool thaw_anim(void *data)
{
    ecore_animator_thaw(data);
    // Animation restart Timer delete
    return ECORE_CALLBACK_CANCEL;
}
```

앱이 실행되고 3초 후에 freeze_anim() 함수가 호출됩니다.

ecore_animator_freeze(Ecore_Animator *) 는 애니메이션을 일시 정지하는 API입니다.

앱이 실행되고 6초 후에 thaw_anim() 함수가 호출됩니다.

`ecore_animator_thaw(Ecore_Animator *)` 는 애니메이션을 재시작하는 API입니다.

예제를 다시 실행하면 3초마다 애니메이션 일시정지와 재시작을 반복합니다.

4) TimeLine 애니메이션

일정한 시간 동안 진행되는 애니메이션을 구현해 보겠습니다.

`create_base_gui()` 함수에 새로운 코드를 추가합니다. 2번째 Rectangle 객체를 생성하고, TimeLine 애니메이션을 생성하는 코드입니다.

```
/* Rect-1 */
ad->rect1 = evas_object_rectangle_add(evas);
evas_object_pass_events_set(ad->rect1, EINA_TRUE);
evas_object_color_set(ad->rect1, 0, 0, 160, 160);
evas_object_resize(ad->rect1, 50, 50);
evas_object_show(ad->rect1);

/* Rect-2 */
ad->rect2 = evas_object_rectangle_add(evas);
evas_object_pass_events_set(ad->rect2, EINA_TRUE);
evas_object_color_set(ad->rect2, 0, 55, 0, 160);
evas_object_resize(ad->rect2, 50, 50);
evas_object_show(ad->rect2);

{
~
}

evas_object_raise(ad->rect1);
```

```

evas_object_raise(ad->rect2);

/* Animation-1 */
Ecore_Animator *anim = ecore_animator_add(on_next_frame1, ad->rect1);
/* add 2 timers to go off every 6 seconds */
ecore_timer_add(6, freeze_anim, anim);
Ecore_Timer *timer = ecore_timer_add(6, thaw_anim, anim);
/* delay the last timer by 3 seconds so the 2 timers are offset */
ecore_timer_delay(timer, 3);

/* Animation-2 */
ecore_animator_timeline_add(4, on_next_frame2, ad->rect2);

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

ecore_animator_timeline_add(double, Ecore_Timeline_Cb, void *) 는 TimeLine 애니메이션을 생성하는 API입니다. 1번째 파라미터는 실행 시간이고 단위는 초입니다. 2번째 파라미터는 프레임 이벤트 함수명, 3번째 파라미터는 사용자 데이터입니다.

2번째 사각형에 위치, 크기, 컬러를 변경하는 애니메이션을 적용해 보겠습니다. create_base_gui() 함수 위에 새로운 코드를 추가합니다. TimeLine 애니메이션 프레임 이벤트 함수입니다.

```

static Eina_Bool on_next_frame2(void *data, double pos)
{
    double frame = ecore_animator_pos_map(pos, pos_map, 1.2, 15);
    evas_object_resize(data, 50 * (1 + frame * 2), 50 * (1 + frame * 2));
    evas_object_move(data, 200 * frame, 200 * frame + 100);
    evas_object_color_set(data, 255 * frame, 0, 255 * (1 - frame), 255);
    return ECORE_CALLBACK_RENEW;
}

```

```
}
```

ecore_animator_pos_map() 는 현재 애니메이션 position에 매핑되는 결과값을 반환하는 API 입니다. 반환되는 값의 범위는 0~1 사이입니다. 애니메이션이 시작할 때는 결과값이 0이었다가 점점 증가해서 애니메이션이 종료될 때는 1이 됩니다. 1번째 파라미터에는 Timeline 애니메이션 이벤트 함수의 2번째 파라미터를 전달하면 됩니다. 2번째 파라미터는 애니메이션 스타일을 지정합니다. 종류는 다음과 같습니다.

- ECORE_POS_MAP_LINEAR, /**< Linear 0.0 -> 1.0 */
- ECORE_POS_MAP_ACCELERATE, /**< Start slow then speed up */
- ECORE_POS_MAP_DECELERATE, /**< Start fast then slow down */
- ECORE_POS_MAP_SINUSOIDAL, /**< Start slow, speed up then slow down at the end */
- ECORE_POS_MAP_ACCELERATE_FACTOR, /**< Start slow then speed up, v1 being a power factor, @c 0.0 being linear, @c 1.0 being normal accelerate, @c 2.0 being much more pronounced accelerate (squared), @c 3.0 being cubed, and so on */
- ECORE_POS_MAP_DECELERATE_FACTOR, /**< Start fast then slow down, v1 being a power factor, @c 0.0 being linear, @c 1.0 being normal decelerate, @c 2.0 being much more pronounced decelerate (squared), @c 3.0 being cubed, and so on */
- ECORE_POS_MAP_SINUSOIDAL_FACTOR, /**< Start slow, speed up then slow down at the end, v1 being a power factor, @c 0.0 being linear, @c 1.0 being normal sinusoidal, @c 2.0 being much more pronounced sinusoidal (squared), @c 3.0 being cubed, and so on */
- ECORE_POS_MAP_DIVISOR_INTERP, /**< Start at gradient * v1, interpolated via power of v2 curve */
- ECORE_POS_MAP_BOUNCE, /**< Start at @c 0.0 then "drop" like a ball

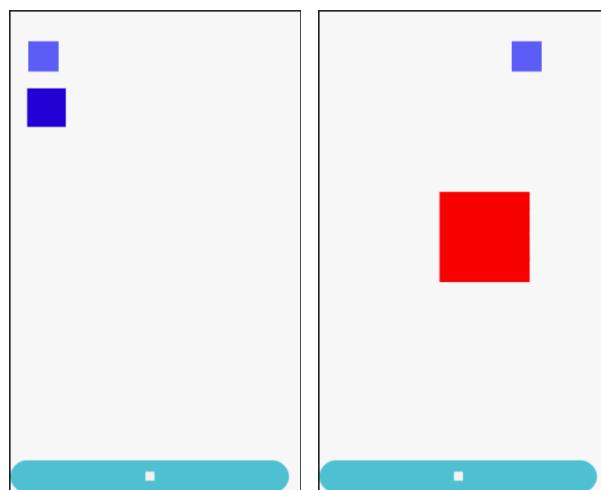
bouncing to the ground at @c 1.0, and bounce v2 times, with decay factor of v1 */

- ECORE_POS_MAP_SPRING /**< Start at @c 0.0 then "wobble" like a spring with rest position @c 1.0, and wobble v2 times, with decay factor of v1 */

ecore_animator_pos_map() 함수의 3번째 파라미터는 속도 변화의 강도를 지정하고, 4번째 파라미터는 속도 변화의 완급을 지정합니다.

그 다음 코드는 사각형의 크기, 위치, 컬러를 지정하는 코드입니다.

예제를 다시 실행하면 파란색 사각형이 우측 아래로 이동하면서 크기는 커지고 컬러는 빨간색으로 변경됩니다.



5) TimeLine 애니메이션에 가속도 적용

방금 만든 애니메이션에 적용된 스타일은 ECORE_POS_MAP_LINEAR입니다. 이것은 일정한 속도를 유지하는 옵션입니다. 처음에는 느리다가 점점 속도가 빨라지는 옵션을 적용해

보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다. 새로운 Button을 생성하는 코드입니다.

```
{  
    /* Button-1 */  
    Evas_Object *btn = elm_button_add(ad->win);  
    elm_object_text_set(btn, "■");  
    evas_object_smart_callback_add(btn, "clicked", btn_stop_cb, NULL);  
    my_table_pack(table, btn, 0, 0, 2, 1);  
  
    /* Button-2 */  
    btn = elm_button_add(ad->win);  
    elm_object_text_set(btn, "Accelerate");  
    evas_object_smart_callback_add(btn, "clicked", btn_accelerate_cb, ad->rect2);  
    my_table_pack(table, btn, 0, 1, 1, 1);  
}  
}
```

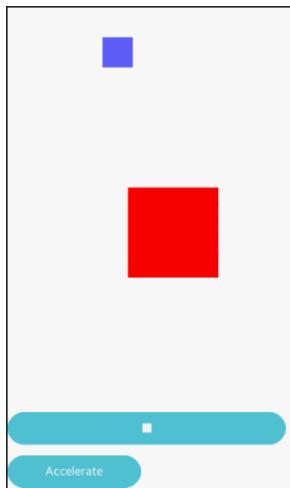
그런 다음 2번째 Button의 콜백 함수를 만들어 보겠습니다.
`create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void  
btn_accelerate_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    pos_map = ECORE_POS_MAP_ACCELERATE;  
    ecore_animator_timeline_add(4, on_next_frame2, data);  
}
```

애니메이션 스타일을 `ECORE_POS_MAP_ACCELERATE` 으로
변경하였습니다. 가속도 옵션입니다.

`ecore_animator_timeline_add()` 함수를 사용해서 Timeline 애니메이션을 생성하였습니다.

예제를 다시 실행하고 Accelerate Button을 눌러봅시다. 애니메이션이 시작됩니다.



6) 그외 애니메이션 스타일

Timeline 애니메이션에 다양한 스타일을 적용해 보겠습니다.
`create_base_gui()` 함수에 4개의 Button을 추가합니다.

```
/* Button-2 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Accelerate");
evas_object_smart_callback_add(btn, "clicked", btn_accelerate_cb, ad->rect2);
my_table_pack(table, btn, 0, 1, 1, 1);

/* Button-3 */
btn = elm_button_add(ad->win);
```

```

elm_object_text_set(btn, "Decelerate");
evas_object_smart_callback_add(btn, "clicked", btn_decelerate_cb, ad->rect2);
my_table_pack(table, btn, 1, 1, 1);

/* Button-4 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Sinusoidal");
evas_object_smart_callback_add(btn, "clicked", btn_sinusoidal_cb, ad->rect2);
my_table_pack(table, btn, 0, 2, 1, 1);

/* Button-5 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Bounce");
evas_object_smart_callback_add(btn, "clicked", btn_bounce_cb, ad->rect2);
my_table_pack(table, btn, 1, 2, 1, 1);

/* Button-6 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Spring");
evas_object_smart_callback_add(btn, "clicked", btn_spring_cb, ad->rect2);
my_table_pack(table, btn, 0, 3, 1, 1);

}

}

```

Button 콜백 함수도 4개가 필요합니다. `create_base_gui()` 함수 위에 새로운 함수 4개를 추가합니다.

```

static void
btn_decelerate_cb(void *data, Evas_Object *obj, void *event_info)
{
    pos_map = ECORE_POS_MAP_DECELERATE;
    ecore_animator_timeline_add(4, on_next_frame2, data);

```

```

}

static void
btn_sinusoidal_cb(void *data, Evas_Object *obj, void *event_info)
{
    pos_map = ECORE_POS_MAP_SINUSOIDAL;
    ecore_animator_timeline_add(4, on_next_frame2, data);
}

static void
btn_bounce_cb(void *data, Evas_Object *obj, void *event_info)
{
    pos_map = ECORE_POS_MAP_BOUNCE;
    ecore_animator_timeline_add(4, on_next_frame2, data);
}

static void
btn_spring_cb(void *data, Evas_Object *obj, void *event_info)
{
    pos_map = ECORE_POS_MAP_SPRING;
    ecore_animator_timeline_add(4, on_next_frame2, data);
}

```

ECORE_POS_MAP_ACCELERATE 는 ECORE_POS_MAP_ACCELERATE의 반대 속성입니다. 처음에 빠르다가 점점 느려집니다.

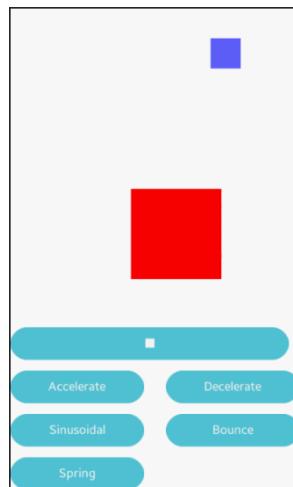
ECORE_POS_MAP_SINUSOIDAL 는 처음에 느리다가 점점 빨라지다가 다시 느려집니다.

ECORE_POS_MAP_BOUNCE 는 공이 튀듯이 진동합니다. 종료 지점을 넘어서지는 않습니다.

ECORE_POS_MAP_SPRING 는 공이 튀듯이 진동합니다. 종료 지정을

넘어섰다가 서서히 멈춥니다.

예제를 다시 실행하고 새로 추가된 Button을 하나씩 차례로 눌러봅시다.



7) 연속 애니메이션

한가지 애니메이션이 종료되면 이어서 2번째 애니메이션이 자동으로 시작되는 기능을 구현해 보겠습니다. Timer를 이용하면 됩니다.
create_base_gui() 함수에서 새로운 Button을 생성합니다.

```
/* Button-6 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Spring");
evas_object_smart_callback_add(btn, "clicked", btn_spring_cb, ad->rect2);
my_table_pack(table, btn, 0, 3, 1, 1);

/* Button-7 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Twice");
evas_object_smart_callback_add(btn, "clicked", btn_twice_cb, ad->rect2);
```

```
    my_table_pack(table, btn, 1, 3, 1, 1);  
}  
}  
}
```

그런 다음 create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

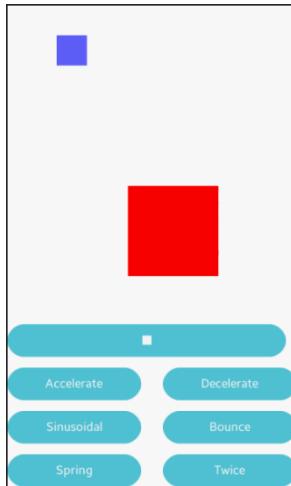
```
static Eina_Bool  
start_second_anim(void *data)  
{  
    pos_map = ECORE_POS_MAP_SPRING;  
    ecore_animator_timeline_add(4, on_next_frame2, data);  
    return ECORE_CALLBACK_CANCEL;  
}  
  
static void  
btn_twice_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    pos_map = ECORE_POS_MAP_ACCELERATE;  
    ecore_animator_timeline_add(4, on_next_frame2, data);  
    ecore_timer_add(4, start_second_anim, data);  
}
```

start_second_anim() 는 2번째 애니메이션을 시작하는 함수입니다.

btn_twice_cb() 는 7번째 Button의 콜백 함수입니다. 가속도 애니메이션을 시작하고, Timer를 이용해서 4초 후에 자동으로 2번째 애니메이션이 시작되게 합니다.

예제를 다시 실행시켜서 'Twice' Button을 눌러봅시다. 가속도

애니메이션이 시작했다가 종료되면 Spring 애니메이션이 시작됩니다.



8) 관련 API

void ecore_animator_frametime_set(double frametime) : 애니메이션 프레임 시간 간격을 지정하는 API. / 파라미터 - 프레임 시간 간격. 단위는 초단위.

Ecore_Animator *ecore_animator_add(Ecore_Task_Cb func, void *data) : Animator 객체를 생성하는 API. / 파라미터 : 프레임 이벤트 콜백 함수, 사용자 데이터. 주로 애니메이션을 적용할 객체 혹은 appdata를 전달합니다.

void ecore_animator_freeze(Ecore_Animator *animator) : 애니메이션을 일시 정지하는 API.

void ecore_animator_thaw(Ecore_Animator *animator) : 애니메이션을 재시작하는 API.

```
Ecore_Animator *ecore_animator_timeline_add(double runtime,  
Ecore_Timeline_Cb func, void *data) : Timeline 애니메이션을 생성하는  
API. / 1번째 파라미터는 실행 시간이고 단위는 초입니다. 2번째  
파라미터는 프레임 이벤트 함수명, 3번째 파라미터는 사용자 데이터  
입니다.
```

```
EAPI double ecore_animator_pos_map(double pos, Ecore_Pos_Map map,  
double v1, double v2) : 현재 애니메이션 position에 매핑되는 결과값을  
반환하는 API. 반환되는 값의 범위는 0~1 사이값. 애니메이션이 시작할  
때는 결과값이 0이었다가 점점 증가해서 애니메이션이 종료될 때는 1이  
됩니다. 1번째 파라미터에는 Timeline 애니메이션 이벤트 함수의 2번째  
파라미터를 전달하면 됩니다. 2번째 파라미터는 Ecore_Pos_Map을  
지정합니다. 3번째 파라미터는 속도 변화의 강도를 지정하고, 4번째  
파라미터는 속도 변화의 완급을 지정합니다.
```

Ecore_Pos_Map의 종류는 다음과 같습니다.

- ECORE_POS_MAP_LINEAR, /**< Linear 0.0 -> 1.0 */
- ECORE_POS_MAP_ACCELERATE, /**< Start slow then speed up */
- ECORE_POS_MAP_DECELERATE, /**< Start fast then slow down */
- ECORE_POS_MAP_SINUSOIDAL, /**< Start slow, speed up then slow
down at the end */
- ECORE_POS_MAP_ACCELERATE_FACTOR, /**< Start slow then speed
up, v1 being a power factor, @c 0.0 being linear, @c 1.0 being normal
accelerate, @c 2.0 being much more pronounced accelerate (squared),
@c 3.0 being cubed, and so on */
- ECORE_POS_MAP_DECELERATE_FACTOR, /**< Start fast then slow
down, v1 being a power factor, @c 0.0 being linear, @c 1.0 being normal
decelerate, @c 2.0 being much more pronounced decelerate (squared),
@c 3.0 being cubed, and so on */

- ECORE_POS_MAP_SINUSOIDAL_FACTOR, /**< Start slow, speed up then slow down at the end, v1 being a power factor, @c 0.0 being linear, @c 1.0 being normal sinusoidal, @c 2.0 being much more pronounced sinusoidal (squared), @c 3.0 being cubed, and so on */
- ECORE_POS_MAP_DIVISOR_INTERP, /**< Start at gradient * v1, interpolated via power of v2 curve */
- ECORE_POS_MAP_BOUNCE, /**< Start at @c 0.0 then "drop" like a ball bouncing to the ground at @c 1.0, and bounce v2 times, with decay factor of v1 */
- ECORE_POS_MAP_SPRING /**< Start at @c 0.0 then "wobble" like a spring with rest position @c 1.0, and wobble v2 times, with decay factor of v1 */

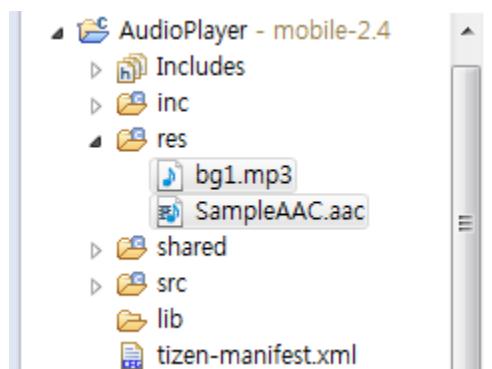
38. Audio 재생

1979년 소니에서 휴대용 오디오 기기를 구상했을 때 시장조사에서는 부정적인 반응이 나왔습니다. 그런데도 소니 오리타 회장은 그대로 밀어붙였고 그 결과 '워크맨'이라는 초대박 밀리언셀러가 탄생했습니다. 21세기에 스티브 잡스는 '아이팟'이라는 작고 간편한 MP3 플레이어를 만들어냈고 전 세계적으로 선풍적인 인기를 끌었습니다. 이제 휴대용 음악 재생기는 스마트폰에 밀려서 점차 역사 속으로 사라지고 있습니다.

Player를 이용하면 오디오와 동영상을 재생할 수 있습니다. 이번 시간에는 오디오 파일을 재생하는 방법을 알아보겠습니다.

1) 오디오 파일 로딩

새로운 소스 프로젝트를 생성하고 Project name을 AudioPlayer 으로 지정합니다. 오디오 파일을 복사해 옵시다. 부록 /Audio 폴더에서 bg1.mp3 와 SampleAAC.aac 파일 2개를 소스 프로젝트 /res 폴더로 복사합니다.



이제 소스코드를 입력해 보겠습니다. src 폴더에 소스파일(~.c)을 열고 화면 위쪽에 라이브러리와 변수를 추가합니다.

```
#include "audioplayer.h"
#include <player.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    player_h player;
} appdata_s;

const char* file_name[] = { "SampleAAC.aac", "bg1.mp3" };
```

player_h 는 오디오, 동영상 같은 미디어를 재생하는 Player 구조체입니다.

file_name[] 는 파일명을 저장하는 문자열 배열입니다.

앱이 실행되면 자동으로 Player 객체를 생성하고 1번째 오디오 파일을 로딩하겠습니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
evas_object_show(ad->win);

// Create Player
ad->player = create_player();

// Load audio file to Player
prepare_player(ad, 0);
```

create_player() 는 Player를 생성하는 함수이고, prepare_player() 는

오디오 파일을 로딩하는 함수입니다. 이제부터 하나씩 만들어 보겠습니다. `create_base_gui()` 함수 위에 6개의 함수를 추가합니다.

```
// Get player state
static player_state_e
get_player_state(player_h player)
{
    player_state_e state;
    player_get_state(player, &state);
    return state;
}

// Play completed event function
static void
on_player_completed(player_h* player)
{
    if(player)
        player_stop(player);
}

// Create Player
static player_h
create_player()
{
    player_h player;

    player_create(&player);
    player_set_completed_cb(player, on_player_completed, player);

    return player;
}

// Stop play
```

```

static void
stop_player(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    if( get_player_state(ad->player) == PLAYER_STATE_PLAYING || get_player_state(ad-
>player) == PLAYER_STATE_PAUSED)
    {
        player_stop(ad->player);
    }
}

// Get full path of source file
static inline const char*
get_resource_path(const char *file_path)
{
    static char absolute_path[PATH_MAX] = {'\0'};

    static char *res_path_buff = NULL;
    if(res_path_buff == NULL)
        res_path_buff = app_get_resource_path();

    snprintf(absolute_path, PATH_MAX, "%s%s", res_path_buff, file_path);
    return absolute_path;
}

// Load file to Player
static void
prepare_player(appdata_s* ad, int index)
{
    // Stop play
    stop_player(ad, NULL, NULL);
    // Close file
    player_unprepare(ad->player);
}

```

```

const char* file = file_name[index];
// Get full path of source file
const char *res_path = get_resource_path(file);

// Load file
player_set_uri(ad->player, res_path);
// Prepare play
int result = player_prepare(ad->player);
dlog_print(DLOG_INFO, "tag", "File load : %d", result);
}

```

`get_player_state()` 는 Player의 현재 상태를 반환하는 함수입니다.

`player_get_state(player_h, player_state_e)` 는 Player의 현재 상태를 반환하는 API입니다. 1번째 파라미터는 Player 객체이고, 2번째 파라미터는 상태값을 반환합니다. 상태종류 `player_state_e`에는 다음과 같은 종류가 있습니다.

- `PLAYER_STATE_NONE`, /**< Player is not created */
- `PLAYER_STATE_IDLE`, /**< Player is created, but not prepared */
- `PLAYER_STATE_READY`, /**< Player is ready to play media */
- `PLAYER_STATE_PLAYING`, /**< Player is playing media */
- `PLAYER_STATE_PAUSED`, /**< Player is paused while playing media */

`on_player_completed()` 는 재생 완료시에 호출되는 콜백 함수입니다.

`player_stop(player_h)` 는 재생을 중지하는 API입니다.

`create_player()` 는 Player 객체를 생성하는 함수입니다.

`player_create(player_h *)` 는 Player 객체를 생성하는 API입니다.

`player_set_completed_cb(player_h, player_completed_cb, void *)` 는 재생 완료시에 호출되는 콜백 함수를 지정하는 API입니다. 2번째 파라미터는 콜백 함수명이고, 3번째 파라미터는 사용자 데이터입니다.

`stop_player()` 는 재생을 중지하는 함수입니다.

`get_resource_path()` 는 /res 폴더에 저장된 파일의 전체 경로를 반환하는 함수입니다.

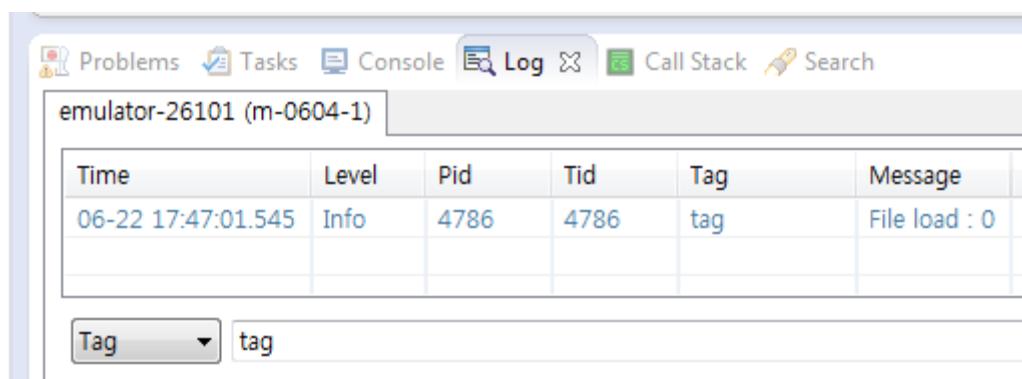
`prepare_player()` 는 오디오 파일을 로딩하는 함수입니다.

`player_unprepare(player_h)` 는 Player에 로딩된 파일을 닫는 API입니다.

`player_set_uri(player_h, const char *)` 는 Player에 파일을 로딩하는 API입니다.

`player_prepare(player_h)` 는 Player가 재생을 준비하는 API입니다. 재생할 준비가 완료되면 0을 반환합니다.

이 상태에서 빌드하고 실행하면 화면에는 아무것도 보이지 않고 소리도 들리지 않습니다. Log 창에서 메시지를 확인하면 'File load : 0'이 보입니다.



2) 오디오 재생

Button을 누르면 오디오 파일을 재생하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double
v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
}
```

```
    evas_object_show(frame);
}


```

create_base_gui() 함수에 새로운 코드를 추가합니다. Box를 생성하고 Button을 추가하는 코드입니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
Evas_Object * box, *btn;

/* A box to put things in vertically - default mode for box */
box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
/* Label*/
ad->label = elm_label_add(ad->win);
elm_object_text_set(ad->label, "<align=center>Hello Tizen</>");
/* expand horizontally but not vertically, and fill horiz,
 * align center vertically */
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.0);
}
```

```

/* Play Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Play");
evas_object_smart_callback_add(btn, "clicked", start_player, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 0.0);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

이제 Button의 콜백 함수를 만들어 봅시다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```

// Start play
static void start_player(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

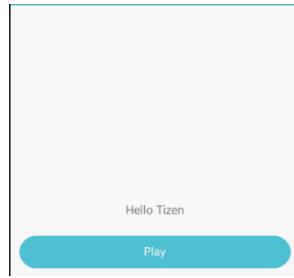
    if( get_player_state(ad->player) != PLAYER_STATE_PLAYING)
        player_start(ad->player);
}

```

`start_player()` 는 오디오 재생을 시작하는 함수입니다.

`player_start(player_h)` 는 오디오 재생을 시작하는 API 입니다.

예제를 다시 실행하고 Play 버튼을 눌러봅시다. 음악이 재생됩니다.



3) 일시정지 & 중지

2개의 Button을 추가해서 일시정지와 중지 기능을 구현해 보겠습니다.
create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Play Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Play");
evas_object_smart_callback_add(btn, "clicked", start_player, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 0.0);

/* Pause Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Pause");
evas_object_smart_callback_add(btn, "clicked", pause_player, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, -1.0);

/* Stop Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Stop");
evas_object_smart_callback_add(btn, "clicked", stop_player, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, -1.0);

}
```

2개의 Button을 추가했습니다. 이제 Button의 콜백 함수를 만들 차례입니다. `create_base_gui()` 함수 위에 새로운 코드를 추가합니다.

```
// Pause play
static void pause_player(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    if( get_player_state(ad->player) == PLAYER_STATE_PLAYING )
        player_pause(ad->player);
}
```

`pause_player()` 는 오디오 재생을 일시정지하는 함수입니다.

`player_pause(player_h)` 는 오디오 재생을 일시정지하는 API 입니다.

Stop Button의 콜백 함수는 따로 만들 필요가 없습니다. 이미 만들어 놓은 `stop_player()` 함수가 있기 때문입니다.

예제를 다시 실행하고 Play 버튼을 누르면 음악이 재생됩니다. Pause 버튼을 누르면 일시정지 되고, 다시 Play 버튼을 누르면 재시작 됩니다. Stop 버튼을 누르면 중지되고, 다시 Play 버튼을 누르면 처음부터 다시 재생됩니다.



4) 오디오 파일 변경

2개의 Button을 추가해서 오디오 파일을 변경하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* Label*/
ad->label = elm_label_add(ad->win);
elm_object_text_set(ad->label, "<align=center>Hello Tizen</>");
/* expand horizontally but not vertically, and fill horiz,
 * align center vertically */
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.0);

/* File Load-1 Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "File-1");
evas_object_smart_callback_add(btn, "clicked", btn_load_file1, ad);
/* epand both horiz and vert, fill horiz and vert */
my_box_pack(box, btn, 1.0, 0.0, -1.0, -1.0);

/* File Load-2 Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "File-2");
evas_object_smart_callback_add(btn, "clicked", btn_load_file2, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, 0.0);

/* Play Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Play");
evas_object_smart_callback_add(btn, "clicked", start_player, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 0.0);
```

2개의 Button을 생성했습니다. 마지막으로 새로 추가된 Button의 콜백

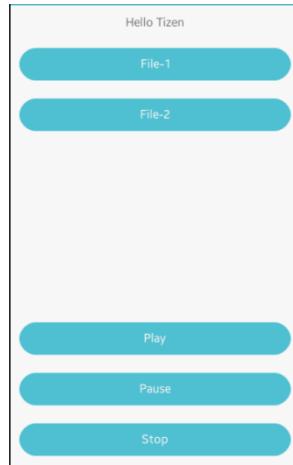
함수를 만들 차례입니다. `create_base_gui()` 함수 위에 새로운 코드를 추가합니다.

```
static void  
btn_load_file1(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    // Load file to Player  
    prepare_player(ad, 0);  
}  
  
static void  
btn_load_file2(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    // Load file to Player  
    prepare_player(ad, 1);  
}
```

`btn_load_file1()` 는 1번째 오디오 파일을 로딩하는 함수입니다.
`prepare_player()` 함수의 2번째 파라미터에 0을 전달하면 1번째 오디오 파일이 로딩됩니다.

`btn_load_file2()` 는 1번째 오디오 파일을 로딩하는 함수입니다.
`prepare_player()` 함수의 2번째 파라미터에 1을 전달하면 2번째 오디오 파일이 로딩됩니다.

예제를 다시 실행시켜서 File-2 버튼을 누른 다음 Play 버튼을 눌러봅시다. 이제까지와 다른 음악이 들립니다. 이번에는 File-1 버튼을 누른 다음 Play 버튼을 눌러봅시다. 먼저번 음악이 들립니다.



5) 관련 API

int player_get_state(player_h player, player_state_e *state) : Player의 현재 상태를 반환하는 API. 1번째 파라미터는 Player 객체이고, 2번째 파라미터는 상태값을 반환합니다. 상태종류 player_state_e에는 다음과 같은 종류가 있습니다.

- PLAYER_STATE_NONE, /**< Player is not created */
- PLAYER_STATE_IDLE, /**< Player is created, but not prepared */
- PLAYER_STATE_READY, /**< Player is ready to play media */
- PLAYER_STATE_PLAYING, /**< Player is playing media */
- PLAYER_STATE_PAUSED, /**< Player is paused while playing media */

int player_stop(player_h player) : 재생을 중지하는 API.

int player_create(player_h *player) : Player 객체를 생성하는 API.

int player_set_sound_type(player_h player, sound_type_e type) : 사운드 종류를 지정하는 API. 오디오 파일을 재생할 때는

SOUND_TYPE_MEDIA를 지정하면 됩니다.

int player_set_volume(player_h player, float left, float right) : 스피커 볼륨을 지정하는 API. 볼륨값의 범위는 0~1.0 사이값입니다. / 파라미터 - Player 객체, 왼쪽 볼륨, 오른쪽 볼륨.

int player_set_looping(player_h player, bool looping) : 반복 재생 여부를 지정하는 API. 2번째 파라미터에 true를 전달하면 반복 재생이고, false를 전달하면 1번만 재생합니다.

int player_set_completed_cb(player_h player, player_completed_cb callback, void *user_data) : 재생 완료시에 호출되는 콜백 함수를 지정하는 API. / 파라미터 - Player 객체, 콜백 함수명, 사용자 데이터.

int player_unprepare(player_h player) : Player에 로딩된 파일을 닫는 API.

int player_set_uri(player_h player, const char * uri) : Player에 파일을 로딩하는 API.

int player_prepare(player_h player) : Player가 재생을 준비하는 API. 재생할 준비가 완료되면 0을 반환합니다.

int player_start(player_h player) : 오디오 재생을 시작하는 API.

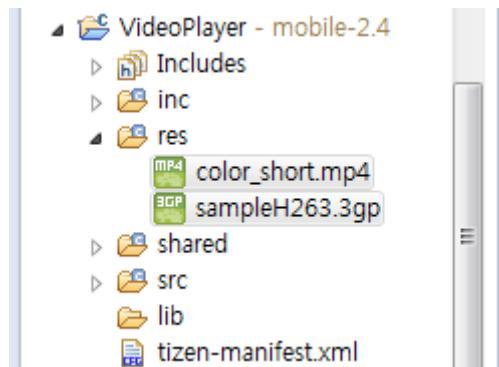
int player_pause(player_h player) : 오디오 재생을 일시정지하는 API.

39. Video 재생

Player를 이용하면 오디오와 동영상을 재생할 수 있습니다. 동영상 재생이 오디오와 다른 점은 동영상을 표시할 스크린이 필요하다는 것입니다. 방법은 이미지 객체를 생성해서 Player와 연동하는 것입니다. 이번 시간에는 비디오 파일을 재생하는 방법을 알아보겠습니다.

1) 스크린 생성

새로운 소스 프로젝트를 생성하고 Project name을 VideoPlayer 으로 지정합니다. 비디오 파일을 복사해 옵시다. 부록 /Video 폴더에서 color_short.mp4 와 sampleH263.3gp 파일 2개를 소스 프로젝트 /res 폴더로 복사합니다.



오디오를 재생할 때는 Player 와 오디오 파일만 있으면 되지만, 비디오를 재생할 때는 스크린이 필요합니다. 스크린을 만드는 과정은 다음과 같습니다.

- EVAS를 생성
- 이미지 객체를 생성
- 이미지 객체를 Player의 Display로 지정

이제 기능을 구현해 보겠습니다. src 폴더에 소스파일(~.c)을 열고 화면 위쪽에 라이브러리와 변수를 추가합니다.

```
#include "videoplayer.h"
#include <player.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    player_h player;
    Evas_Object *video_rect;
} appdata_s;

const char* file_name[] = { "sampleH263.3gp", "color_short.mp4" };
```

player_h 는 오디오, 동영상 같은 미디어를 재생하는 Player 구조체입니다. video_rect 는 동영상이 표시되는 Image 객체입니다.

file_name[] 는 파일명을 저장하는 문자열 배열입니다.

스크린으로 사용할 Image 객체와 화면 배경으로 사용할 Bg 위젯을 생성하겠습니다. 동영상을 제대로 감상하려면 배경이 검정색이어야 하기 때문입니다. create_base_gui() 함수 위에 새로운 함수를 생성합니다. Table에 위젯을 추가하는 함수입니다.

```
static void
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)
{
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
```

```
    elm_table_pack(table, child, x, y, w, h);
    evas_object_show(child);
}

-----
```

그런 다음 `create_base_gui()` 함수에 새로운 코드를 추가합니다.
Conformant와 Label 생성코드는 주석 처리합니다.

```
-----  
/* Conformant */  
/*ad->conform = elm_conformant_add(ad->win);  
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);  
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);  
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
elm_win_resize_object_add(ad->win, ad->conform);  
evas_object_show(ad->conform);*/  
  
/* Label*/  
/*ad->label = elm_label_add(ad->conform);  
elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");  
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
elm_object_content_set(ad->conform, ad->label);*/  
  
Evas_Object *bg, *btn;  
  
/* Background */  
bg = elm_bg_add(ad->win);  
elm_bg_color_set(bg, 0, 0, 0);  
elm_win_resize_object_add(ad->win, bg);  
evas_object_show(bg);  
  
{  
/* Box to put the table in so we can bottom-align the table
```

```

        /* window will stretch all resize object content to win size */
        Evas_Object *box = elm_box_add(ad->win);
        evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
        elm_win_resize_object_add(ad->win, box);
        evas_object_show(box);

/* Table */
Evas_Object *table = elm_table_add(ad->win);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 10 * elm_config_scale_get(), 10 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fiill width but align to bottom of box */
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_box_pack_end(box, table);
evas_object_show(table);

{
    /* Container: standard table */
    Evas_Object *tbl = elm_table_add(ad->win);
    evas_object_size_hint_weight_set(tbl, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(tbl, EVAS_HINT_FILL, EVAS_HINT_FILL);
    elm_object_content_set(ad->conform, tbl);
    evas_object_show(tbl);

    /* The video object */
    ad->video_rect = video_rect_add(tbl);
    my_table_pack(table, ad->video_rect, 0, 0, 3, 3);
}

```

```
}

/* Show window after base gui is set up */
evas_object_show(ad->win);
```

배경 컬러를 검정색으로 변경하기 위해서 Bg 위젯을 생성하였습니다.
위젯을 비율단위로 배치하기 위해서 Tabel 컨테이너를 생성하였고, 위젯
사이의 간격을 지정하기 위해서 Box를 사용하였습니다.

video_rect_add() 는 Evas 이미지 객체를 생성해서 반환하는 함수입니다.
이제부터 이 함수를 만들어 보겠습니다. create_base_gui() 함수 위에
새로운 함수 2개를 추가합니다.

```
// Get full path of resource file
static inline const char *get_resource_path(const char *file_path)
{
    static char absolute_path[PATH_MAX] = {'\0'};
    static char *res_path_buff = NULL;
    if(res_path_buff == NULL)
    {
        res_path_buff = app_get_resource_path();
    }

    sprintf(absolute_path, PATH_MAX, "%s%s", res_path_buff, file_path);
    return absolute_path;
}

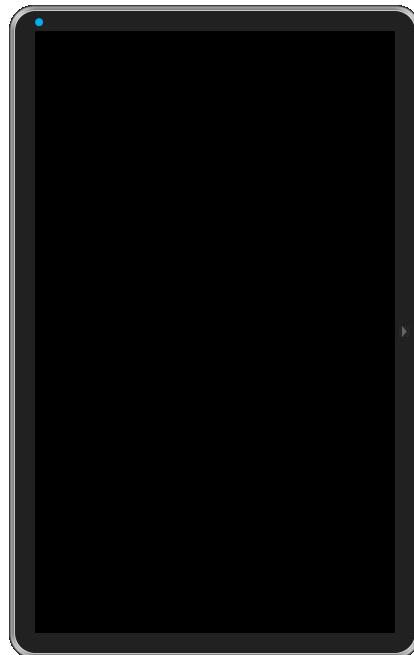
// Create Image for screen
static Evas_Object *
video_rect_add(Evas_Object *parent)
{
    Evas *evas = evas_object_evas_get(parent);
```

```
Evas_Object *image = evas_object_image_filled_add(evas);
evas_object_size_hint_weight_set(image, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(image, EVAS_HINT_FILL, EVAS_HINT_FILL);
evas_object_show(image);
return image;
}
```

get_resource_path() 는 /res 폴더에 저장된 파일의 전체 경로를 반환하는 함수입니다. 잠시 후에 동영상 파일을 로딩할 때 사용하겠습니다.

video_rect_add() 는 스크린 영역에 Evas 객체와 Image 객체를 생성하는 함수입니다.

예제를 빌드하고 실행시켜 봅시다. 화면 전체가 검정색으로 보입니다.



2) 비디오 파일 로딩

앱이 실행되면 자동으로 Player 객체를 생성하고 1번째 비디오 파일을 로딩하겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

// Create Player
ad->player = create_player();

// Load audio file to Player
prepare_player(ad, 0);

}
```

`create_player()` 는 Player를 생성하는 함수이고, `prepare_player()` 는 비디오 파일을 로딩하는 함수입니다. 이제부터 하나씩 만들어 보겠습니다. `create_base_gui()` 함수 위에 5개의 함수를 추가합니다. 여기서 부터는 AudioPlayer 예제와 거의 유사합니다.

```
// Get player state
static player_state_e
get_player_state(player_h player)
{
    player_state_e state;
    player_get_state(player, &state);
    return state;
}

// Play completed event function
```

```

static void
on_player_completed(player_h* player)
{
    if(player)
        player_stop(player);
}

// Create player
static player_h
create_player()
{
    player_h player;

    player_create(&player);
    player_set_sound_type(player, SOUND_TYPE_MEDIA);
    player_set_volume(player, 1.0, 1.0);
    player_set_looping(player, false);
    player_set_completed_cb(player, on_player_completed, player);

    return player;
}

// Stop play
static void
stop_player(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    if( get_player_state(ad->player) == PLAYER_STATE_PLAYING || get_player_state(ad->player) == PLAYER_STATE_PAUSED)
        player_stop(ad->player);
}

// Load video file to Player
static void

```

```

prepare_player(appdata_s* ad, int index)
{
    player_stop(ad->player);
    // Close file
    player_unprepare(ad->player);

    const char* file = file_name[index];
    // Get full path of resource file
    const char *res_path = get_resource_path(file);

    // Load file
    player_set_uri(ad->player, res_path);
    player_set_display(ad->player,     PLAYER_DISPLAY_TYPE_EVAS,     GET_DISPLAY(ad-
>video_rect));
    player_set_display_mode(ad->player, PLAYER_DISPLAY_MODE_FULL_SCREEN);
    // Prepare play
    int result = player_prepare(ad->player);
    dlog_print(DLOG_INFO, "tag", "File load : %d", result);
}

```

get_player_state() 는 Player의 현재 상태를 반환하는 함수입니다.

player_get_state(player_h, player_state_e) 는 Player의 현재 상태를 반환하는 API입니다. 1번째 파라미터는 Player 객체이고, 2번째 파라미터는 상태값을 반환합니다. 상태종류 player_state_e에는 다음과 같은 종류가 있습니다.

- PLAYER_STATE_NONE, /**< Player is not created */
- PLAYER_STATE_IDLE, /**< Player is created, but not prepared */
- PLAYER_STATE_READY, /**< Player is ready to play media */
- PLAYER_STATE_PLAYING, /**< Player is playing media */
- PLAYER_STATE_PAUSED, /**< Player is paused while playing media */

`on_player_completed()` 는 재생 완료시에 호출되는 콜백 함수입니다.

`player_stop(player_h)` 는 재생을 중지하는 API입니다.

`create_player()` 는 Player 객체를 생성하는 함수입니다.

`player_create(player_h *)` 는 Player 객체를 생성하는 API입니다.

`player_set_sound_type(player_h, sound_type_e)` 는 사운드 종류를 지정하는 API입니다. 비디오 파일을 재생할 때는 `SOUND_TYPE_MEDIA`를 지정하면 됩니다.

`player_set_volume(player_h, float, float)` 는 스피커 볼륨을 지정하는 API입니다. 볼륨값의 범위는 0~1.0 사이값입니다. 2번째 파라미터는 왼쪽 볼륨, 3번째 파라미터는 오른쪽 볼륨 입니다.

`player_set_looping(player_h, bool)` 는 반복 재생 여부를 지정하는 API입니다. 2번째 파라미터에 `true`를 전달하면 반복 재생이고, `false`를 전달하면 1번만 재생합니다.

`player_set_completed_cb(player_h, player_completed_cb, void *)` 는 재생 완료시에 호출되는 콜백 함수를 지정하는 API입니다. 2번째 파라미터는 콜백 함수명이고, 3번째 파라미터는 사용자 데이터입니다.

`stop_player()` 는 재생을 중지하는 함수입니다.

`prepare_player()` 는 비디오 파일을 로딩하는 함수입니다.

`player_unprepare(player_h)` 는 Player에 로딩된 파일을 닫는 API입니다.

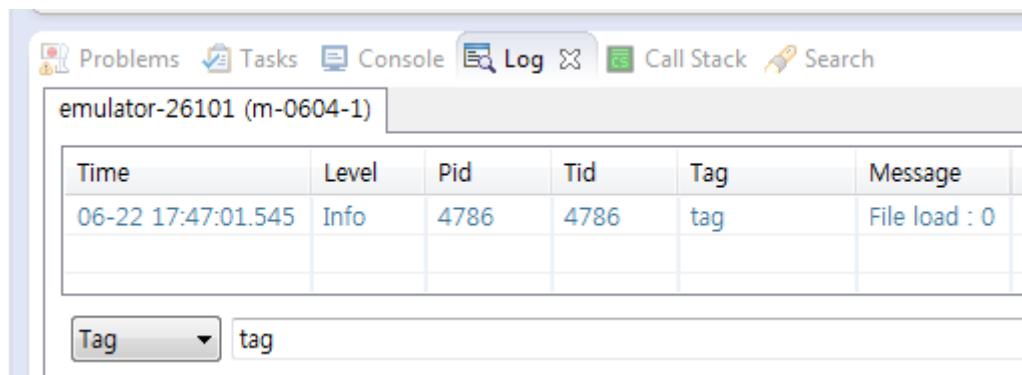
`player_set_uri(player_h, const char *)` 는 Player에 파일을 로딩하는 API입니다.

`player_set_display(player_h, player_display_type_e, player_display_h)` 는 Player에 스크린을 지정하는 함수입니다. 1번째 파라미터는 Player 객체, 2번째는 스크린 종류를 지정합니다. 캔버스 기반인 경우에는 `PLAYER_DISPLAY_TYPE_EVAS`를 지정하면 됩니다. 3번째 파라미터는 스크린에 해당하는 `player_display_h` 객체를 전달합니다.

`GET_DISPLAY()` 는 객체에서 `player_display_h`를 구하는 함수입니다.

`player_prepare(player_h)` 는 Player가 재생을 준비하는 API입니다. 재생할 준비가 완료되면 0을 반환합니다.

이 상태에서 빌드하고 실행하면 화면에는 아무것도 보이지 않고 소리도 들리지 않습니다. Log 창에서 메시지를 확인하면 'File load : 0'이 보입니다.



The screenshot shows the Eclipse IDE's Log view window. The title bar includes tabs for Problems, Tasks, Console, Log, Call Stack, and Search. The main area displays a single log entry for the emulator process:

Time	Level	Pid	Tid	Tag	Message
06-22 17:47:01.545	Info	4786	4786	tag	File load : 0

Below the table, there is a dropdown menu labeled "Tag" with the value "tag" selected.

3) 비디오 재생

Button을 누르면 비디오 파일을 재생하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Table */
```

```

~  

elm_box_pack_end(box, table);  

evas_object_show(table);  
  

{  

/* Play Button */  

btn = elm_button_add(ad->win);  

elm_object_text_set(btn, "Play");  

evas_object_smart_callback_add(btn, "clicked", start_player, (void*)ad);  

my_table_pack(table, btn, 0, 3, 1, 1);  
  

/* Container: standard table */  

Evas_Object *tbl = elm_table_add(ad->win);  

evas_object_size_hint_weight_set(tbl, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);  

evas_object_size_hint_align_set(tbl, EVAS_HINT_FILL, EVAS_HINT_FILL);  

elm_object_content_set(ad->conform, tbl);  

evas_object_show(tbl);

```

Button을 생성하는 코드를 추가하였습니다. Button의 콜백 함수를 만들어 봅시다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```

// Start play  

static void  

start_player(void *data, Evas_Object *obj, void *event_info)  

{  

    appdata_s *ad = data;  
  

    if( get_player_state(ad->player) != PLAYER_STATE_PLAYING)  

        player_start(ad->player);  

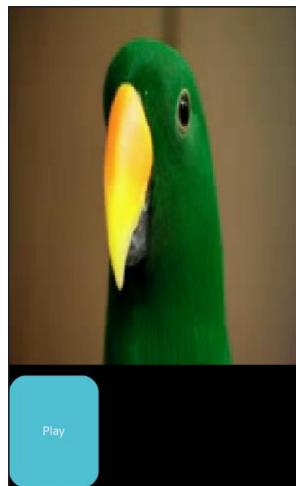
}

```

`start_player()` 는 비디오 재생을 시작하는 함수입니다.

`player_start(player_h)` 는 비디오 재생을 시작하는 API 입니다.

예제를 다시 실행하고 Play 버튼을 눌러봅시다. 동영상이 재생됩니다.



4) 일시정지 & 중지

2개의 Button을 추가해서 일시정지와 중지 기능을 구현해 보겠습니다.
`create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* Play Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Play");
evas_object_smart_callback_add(btn, "clicked", start_player, (void*)ad);
my_table_pack(table, btn, 0, 3, 1, 1);

/* Pause Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Pause");
```

```

evas_object_smart_callback_add(btn, "clicked", pause_player, (void*)ad);
my_table_pack(table, btn, 1, 3, 1, 1);

/* Stop Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Stop");
evas_object_smart_callback_add(btn, "clicked", stop_player, (void*)ad);
my_table_pack(table, btn, 2, 3, 1, 1);

/* Container: standard table */
Evas_Object *tbl = elm_table_add(ad->win);
evas_object_size_hint_weight_set(tbl, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(tbl, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(ad->conform, tbl);
evas_object_show(tbl);

```

2개의 Button을 추가했습니다. 이제 Button의 콜백 함수를 만들 차례입니다. `create_base_gui()` 함수 위에 새로운 코드를 추가합니다.

```

// Pause play
static void pause_player(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    if( get_player_state(ad->player) == PLAYER_STATE_PLAYING )
        player_pause(ad->player);
}

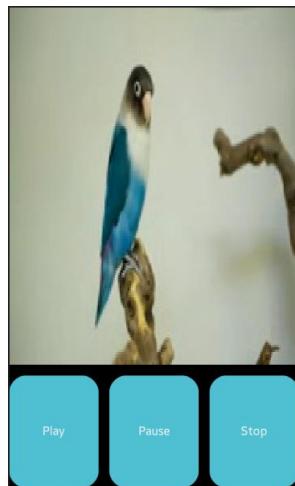
```

`pause_player()` 는 비디오 재생을 일시정지하는 함수입니다.

`player_pause(player_h)` 는 비디오 재생을 일시정지하는 API 입니다.

Stop Button의 콜백 함수는 따로 만들 필요가 없습니다. 이미 만들어 놓은 stop_player() 함수가 있기 때문입니다.

예제를 다시 실행하고 Play 버튼을 누르면 동영상이 재생됩니다. Pause 버튼을 누르면 일시정지 되고, 다시 Play 버튼을 누르면 재시작 됩니다. Stop 버튼을 누르면 중지되고, 다시 Play 버튼을 누르면 처음부터 다시 재생됩니다.



5) 비디오 파일 변경

2개의 Button을 추가해서 비디오 파일을 변경하는 기능을 구현해 보겠습니다. create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Stop Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Stop");
evas_object_smart_callback_add(btn, "clicked", stop_player, (void*)ad);
my_table_pack(table, btn, 2, 3, 1, 1);
```

```

/* File Load-1 Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "File-1");
evas_object_smart_callback_add(btn, "clicked", btn_load_file1, (void*)ad);
my_table_pack(table, btn, 0, 4, 3, 1);

/* File Load-2 Button */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "File-2");
evas_object_smart_callback_add(btn, "clicked", btn_load_file2, (void*)ad);
my_table_pack(table, btn, 0, 5, 3, 1);

/* Container: standard table */
Evas_Object *tbl = elm_table_add(ad->win);
evas_object_size_hint_weight_set(tbl, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(tbl, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(ad->conform, tbl);
evas_object_show(tbl);

```

2개의 Button을 생성했습니다. 마지막으로 새로 추가된 Button의 콜백 함수를 만들 차례입니다. `create_base_gui()` 함수 위에 새로운 코드를 추가합니다.

```

static void
btn_load_file1(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    // Load file to Player
    prepare_player(ad, 0);
}

static void

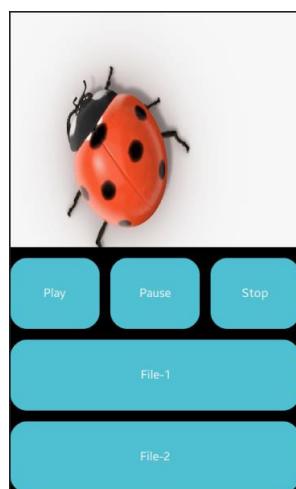
```

```
btn_load_file2(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    // Load file to Player
    prepare_player(ad, 1);
}
```

btn_load_file1() 는 1번째 비디오 파일을 로딩하는 함수입니다.
prepare_player() 함수의 2번째 파라미터에 0을 전달하면 1번째 비디오 파일이 로딩됩니다.

btn_load_file2() 는 1번째 비디오 파일을 로딩하는 함수입니다.
prepare_player() 함수의 2번째 파라미터에 1을 전달하면 2번째 비디오 파일이 로딩됩니다.

예제를 다시 실행시켜서 File-2 버튼을 누른 다음 Play 버튼을 눌러봅시다. 이제까지와 다른 영상이 보입니다. 이번에는 File-1 버튼을 누른 다음 Play 버튼을 눌러봅시다. 먼저번 영상이 보입니다.



6) 관련 API

int player_get_state(player_h player, player_state_e *state) : Player의 현재 상태를 반환하는 API. 1번째 파라미터는 Player 객체이고, 2번째 파라미터는 상태값을 반환합니다. 상태종류 player_state_e에는 다음과 같은 종류가 있습니다.

- PLAYER_STATE_NONE, /**< Player is not created */
- PLAYER_STATE_IDLE, /**< Player is created, but not prepared */
- PLAYER_STATE_READY, /**< Player is ready to play media */
- PLAYER_STATE_PLAYING, /**< Player is playing media */
- PLAYER_STATE_PAUSED, /**< Player is paused while playing media */

int player_stop(player_h player) : 재생을 중지하는 API.

int player_create(player_h *player) : Player 객체를 생성하는 API.

int player_set_sound_type(player_h player, sound_type_e type) : 사운드 종류를 지정하는 API. 비디오 파일을 재생할 때는 SOUND_TYPE_MEDIA를 지정하면 됩니다.

int player_set_volume(player_h player, float left, float right) : 스피커 볼륨을 지정하는 API. 볼륨값의 범위는 0~1.0 사이값입니다. / 파라미터 - Player 객체, 왼쪽 볼륨, 오른쪽 볼륨.

int player_set_looping(player_h player, bool looping) : 반복 재생 여부를 지정하는 API. 2번째 파라미터에 true를 전달하면 반복 재생이고, false를 전달하면 1번만 재생합니다.

int player_set_completed_cb(player_h player, player_completed_cb callback, void *user_data) : 재생 완료시에 호출되는 콜백 함수를

지정하는 API. / 파라미터 - Player 객체, 콜백 함수명, 사용자 데이터.

int player_unprepare(player_h player) : Player에 로딩된 파일을 닫는 API.

int player_set_uri(player_h player, const char * uri) : Player에 파일을 로딩하는 API.

int player_prepare(player_h player) : Player가 재생을 준비하는 API. 재생할 준비가 완료되면 0을 반환합니다.

int player_start(player_h player) : 비디오 재생을 시작하는 API.

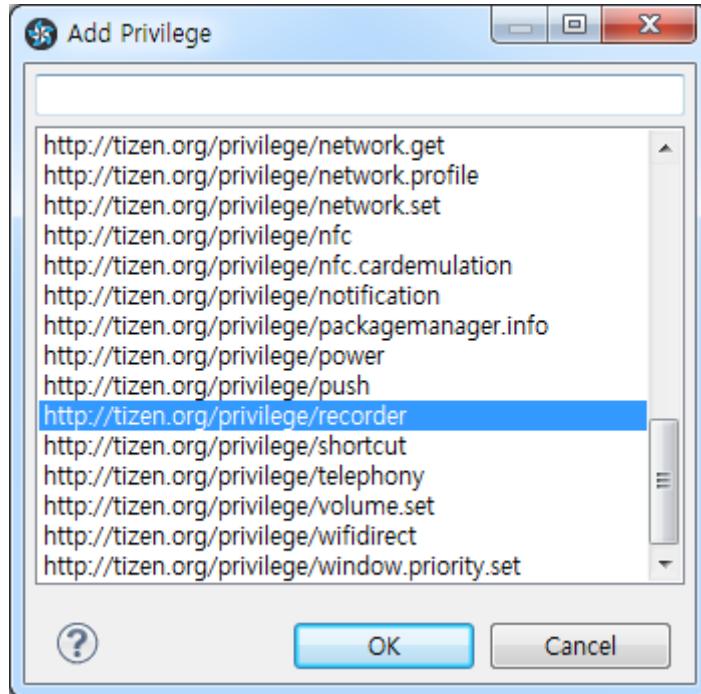
int player_pause(player_h player) : 비디오 재생을 일시정지하는 API.

40. Audio 녹음

스마트폰에 내장되어 있는 마이크를 이용해서 녹음을 할 수 있습니다. 공연장에서 음악을 녹음하거나, 자신의 목소리를 녹음해서 문자메시지 대신 다른 사람에게 보낼 수도 있고, 메모장으로도 활용할 수 있습니다. 예제를 통해서 구체적인 방법을 알아보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 RecorderEx 으로 지정합니다. Recorder를 사용하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/recorder>를 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



저장하고 아래쪽 텁버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest      xmlns="http://tizen.org/ns/packages"      api-version="2.3"
package="org.example.recorderex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.recorderex" exec="recorderex" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>recorderex</label>
        <icon>recorderex.png</icon>
    </ui-application>
    <privileges>
        <b><privilege>http://tizen.org/privilege/recorder</privilege></b>
    </privileges>
</manifest>
```

2) Recorder 생성

Recorder를 생성하기 위해서는 Codec 과 파일 형식, 그리고 품질 등을 지정해야 합니다. src 폴더에 소스파일(~.c)을 열고 화면 위쪽에 라이브러리와 변수, 그리고 구조체 등을 추가합니다.

```
#include "recorderex.h"
#include <recorder.h>
#include <player.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;

    Evas_Object *btn_rec, *btn_recstop, *btn_play, *btn_playstop;

    player_h player;
    recorder_h recorder;
    recorder_audio_codec_e *codec_list;
    int codec_list_len;
    char file_path[PATH_MAX];
    recorder_audio_codec_e codec;
    recorder_file_format_e file_format;
    FILE *preproc_file;
} appdata_s;

typedef struct
{
    recorder_audio_codec_e *codec_list;
    int len;
} supported_encoder_data;
```

이번 예제에는 총 4개의 Button 위젯을 생성할 것입니다. 각 Button의 Enable 상태를 변경하기 위해서 4개의 변수(btn_rec, btn_recstop, btn_play, btn_playstop)를 선언하였습니다.

player_h 는 AudioPlayer와 VideoPlayer 예제에서도 사용했던 재생기 구조체 입니다.

recorder_h 는 녹음기 구조체 입니다.

recorder_audio_codec_e 는 코덱 종류를 저장하는 Enumeration 입니다.

codec_list_len 는 코덱 목록 개수를 저장하는 변수입니다.

file_path[] 는 녹음 파일 경로를 저장하는 문자열 변수입니다.

recorder_file_format_e 는 파일 형식 정보를 저장하는 Enumeration 입니다.

FILE는 파일 입출력에 사용하는 파일 데이터 스트림 입니다.

supported_encoder_data 는 지원되는 코덱 목록을 저장하는 구조체 입니다.

Recorder 객체를 생성하겠습니다. create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

// Create recorder
_recorder_create(ad);
ad->codec_list = audio_recorder_get_supported_encoder(ad->recorder, &ad-
```

```

>codec_list_len);
ad->codec      =      ad->codec_list_len      ?      ad->codec_list[0]      :
RECODER_AUDIO_CODEC_PCM;

_codec_set(ad, codec);
}

```

_recorder_create() 는 Recorder를 생성하는 함수입니다.

audio_recorder_get_supported_encoder() 는 지원되는 Codec 목록을 구해서 반환하는 함수입니다.

코덱 목록을 구했으면 1번째 코덱을 codec 이라는 변수에 저장하고 목록이 존재하지 않는다면 PCM 코덱을 저장합니다.

_codec_set() 는 코덱을 지정하는 함수입니다.

Recorder 생성에 필요한 코드를 입력하겠습니다. create_base_gui() 함수 위에 새로운 함수 8개를 추가합니다.

```

// Check is recording
static bool
_recorder_is_recording(appdata_s *ad)
{
    recorder_state_e state = RECODER_STATE_NONE;
    recorder_get_state(ad->recorder, &state);
    return state == RECODER_STATE_RECORDING;
}

// Stop recording

```

```

static void
record_stop(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    if (ad->recorder)
    {
        recorder_commit(ad->recorder);
        // Check is recording
        if (!recorder_is_recording(ad))
        {
            recorder_unprepare(ad->recorder);
        }
        elm_object_disabled_set(ad->btn_play, EINA_FALSE);
        elm_object_disabled_set(ad->btn_rec, EINA_FALSE);
        elm_object_disabled_set(ad->btn_playstop, EINA_TRUE);
        elm_object_disabled_set(ad->btn_recstop, EINA_TRUE);
    }
}

// Maximum recording time event callback function
static void
_on_recording_limit_reached_cb(recorder_recording_limit_type_e type, void *user_data)
{
    appdata_s *ad = user_data;
    if(type == RECORDER_RECORDING_LIMIT_TIME)
        // Stop recording
        record_stop(ad, NULL, NULL);
}

// Create recorder
static void
_recorder_create(appdata_s *ad)
{
    if(recorder_create_audiorecorder(&ad->recorder) == RECORDER_ERROR_NONE)
    {

```

```

// Set maximum recording time event callback function
recorder_set_recording_limit_reached_cb(ad->recorder,
_on_recording_limit_reached_cb, ad);
    recorder_attr_set_audio_channel(ad->recorder, 1);
    recorder_attr_set_audio_device(ad->recorder, RECORDER_AUDIO_DEVICE_MIC);
    recorder_attr_set_time_limit(ad->recorder, 20);
}
}

static bool
_recorder_supported_audio_encoder_cb(recorder_audio_codec_e codec, void *user_data)
{
    bool result = false;
    supported_encoder_data *data = user_data;

    if(data && codec != RECORDER_AUDIO_CODEC_DISABLE)
    {
        data->codec_list = realloc(data->codec_list, sizeof(supported_encoder_data) *
(data->len + 1));
        data->codec_list[data->len] = codec;
        ++(data->len);
        result = true;
    }

    return result;
}

recorder_audio_codec_e*
audio_recorder_get_supported_encoder(recorder_h recorder, int *list_length)
{
    supported_encoder_data data = {0};
    data.codec_list = NULL;
    data.len = 0;

    int res = recorder_FOREACH_SUPPORTED_AUDIO_ENCODER(recorder,

```

```

_recorder_supported_audio_encoder_cb, &data);

if(res && list_length)
{
    *list_length = data.len;
}

return data.codec_list;
}

const char*
get_file_format_by_codec(appdata_s* ad)
{
    switch(ad->codec)
    {
        case RECORDER_AUDIO_CODEC_AMR:
            ad->file_format = RECORDER_FILE_FORMAT_AMR;
            return "AMR";
            break;

        case RECORDER_AUDIO_CODEC_AAC:
            ad->file_format = RECORDER_FILE_FORMAT_MP4;
            return "MP4";
            break;

        case RECORDER_AUDIO_CODEC_VORBIS:
            ad->file_format = RECORDER_FILE_FORMAT_OGG;
            return "OGG";
            break;
    }

    ad->file_format = RECORDER_FILE_FORMAT_WAV;
    return "WAV";
}

static void
_codec_set(appdata_s *ad)

```

```
{  
    char file_name[NAME_MAX] = {'\0'};  
    const char *file_ext = get_file_format_by_codec(ad);  
  
    char *data_path = app_get_data_path();  
    snprintf(file_name, NAME_MAX, "record.%s", file_ext);  
    snprintf(ad->file_path, PATH_MAX, "%s%s", data_path, file_name);  
    free(data_path);  
}  
_____
```

_recorder_is_recording() 는 녹음 상태를 체크하는 함수입니다.

record_stop() 는 녹음을 중지하는 함수입니다.

recorder_commit(recorder_h recorder) 는 녹음을 중지하고 녹음 데이터를 저장하는 API입니다

recorder_unprepare(recorder_h recorder) 는 녹음기를 초기화하는 API입니다

_on_recording_limit_reached_cb() 는 녹음 최대 시간에 도달했을 때 호출되는 이벤트 함수입니다. 녹음을 강제 중지합니다.

_recorder_create() 는 Recorder를 생성하는 함수입니다.

recorder_create_audiorecorder() 는 Recorder를 생성하는 API 입니다.

recorder_set_recording_limit_reached_cb() 는 녹음 최대 시간 도달 이벤트 함수를 지정하는 API 입니다.

recorder_attr_set_audio_channel() 는 오디오 채널 개수를 지정하는 API 입니다. Mono 인 경우에는 1을 지정하고, Stereo 인 경우에는 2를

지정하면 됩니다.

recorder_attr_set_audio_device() 는 녹음 장비를 지정하는 API 입니다.

스마트폰에서는 마이크를 사용하기 때문에

RECORDER_AUDIO_DEVICE_MIC를 지정하면 됩니다.

recorder_attr_set_time_limit() 녹음 최대 시간을 지정하는 API 입니다.

단위는 초입니다.

_recorder_supported_audio_encoder_cb() 는 지원되는 오디오 코덱 수신 이벤트 콜백 함수입니다.

audio_recorder_get_supported_encoder() 는 지원되는 코덱 목록을 구해서 반환하는 함수입니다.

recorder_FOREACH_SUPPORTED_AUDIO_ENCODER(recorder_h recorder,
recorder_SUPPORTED_AUDIO_ENCODER_CB callback, void *user_data) 는
지원되는 코덱 목록을 요청하는 API입니다. 곧바로 목록을 구하는 것이
아니라 콜백 함수에 목록 데이터가 전달됩니다.

get_file_format_by_codec() 는 코덱 종류에 따라서 파일 포맷과 파일 확장자명을 반환하는 함수입니다.

_codec_set() 는 코덱을 지정하는 함수입니다.

3) 녹음 시작

화면에 2개의 Button을 추가해서 녹음 시작과 녹음 중지 기능을 구현해 보겠습니다. create_base_gui() 함수에 새로운 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

Evas_Object *btn, *frame, *tbl;

/* Frame for some outer padding */
frame = elm_frame_add(ad->conform);
elm_object_style_set(frame, "pad_medium");
elm_object_content_set(ad->conform, frame);
evas_object_show(frame);

/* Table to pack our elements */
tbl = elm_table_add(frame);
elm_table_padding_set(tbl, 5 * elm_scale_get(), 5 * elm_scale_get());
elm_object_content_set(frame, tbl);
evas_object_show(tbl);

{

/* Just a label */
ad->label = elm_label_add(tbl);
elm_object_text_set(ad->label, "Audio recorder");
evas_object_size_hint_align_set(ad->label, 0.5, 0.5);
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_table_pack(tbl, ad->label, 0, 0, 2, 1);
evas_object_show(ad->label);
}
}

```

```

/* Record Start */
btn = elm_button_add(tbl);
elm_object_text_set(btn, "Recording Start");
evas_object_smart_callback_add(btn, "clicked", record_start, (void*)ad);
evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.5);
elm_table_pack(tbl, btn, 0, 1, 1, 1);
evas_object_show(btn);
ad->btn_rec = btn;

/* Record Stop */
btn = elm_button_add(tbl);
elm_object_disabled_set(btn, EINA_TRUE);
elm_object_text_set(btn, "Recording Stop");
evas_object_smart_callback_add(btn, "clicked", record_stop, (void*)ad);
evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.5);
elm_table_pack(tbl, btn, 1, 1, 1, 1);
evas_object_show(btn);
ad->btn_recstop = btn;
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

비율 단위로 위젯을 배치하기 위해서 Table 컨테이너를 생성하였고 Frame 사용해서 바깥쪽 여백을 지정하였습니다.

녹음 시작 기능을 구현해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수 2개를 추가합니다.

```

// Apply settings to recorder
static void _recorder_apply_settings(appdata_s *ad)
{
    if(ad->recorder)
    {
        // Set record file name
        recorder_set_filename(ad->recorder, ad->file_path);
        // Set record file format
        recorder_set_file_format(ad->recorder, ad->file_format);
        // Set record codec
        recorder_set_audio_encoder(ad->recorder, ad->codec);
    }
}

// Start recording
static void
record_start(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    if (ad->recorder)
    {
        // Apply settings to recorder
        _recorder_apply_settings(ad);
        recorder_prepare(ad->recorder);
        recorder_start(ad->recorder);
        elm_object_disabled_set(ad->btn_recstop, EINA_FALSE);
        elm_object_disabled_set(ad->btn_rec, EINA_TRUE);
        elm_object_disabled_set(ad->btn_play, EINA_TRUE);
        elm_object_disabled_set(ad->btn_playstop, EINA_TRUE);
    }
}

```

`_recorder_apply_settings()` 는 녹음 정보를 지정하는 함수입니다.

recorder_set_filename() 는 녹음 파일명을 지정하는 API 입니다.

recorder_set_file_format() 는 녹음 포맷을 지정하는 API입니다.

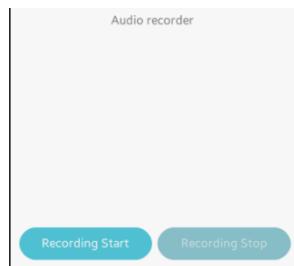
recorder_set_audio_encoder() 는 코덱을 지정하는 API입니다.

record_start() 는 녹음을 시작하는 함수입니다.

recorder_prepare() 는 녹음을 준비하는 API입니다.

recorder_start() 는 녹음을 시작하는 API입니다.

예제를 빌드하고 실행해서 Rec Start 버튼을 누르면 녹음이 시작됩니다. 20초가 지나면 자동으로 녹음이 종료됩니다. Rec Stop 버튼을 누르면 수동으로 중지할 수 있습니다. 안타깝게도 녹음된 오디오 파일을 들어볼 수는 없습니다. 오디오 재생 기능이 아직 구현되지 않았기 때문입니다.



4) 녹음 파일 재생

녹음된 오디오 파일을 재생하는 기능을 구현해 보겠습니다.

create_base_gui() 함수에 새로운 Button 2개를 생성하는 코드를 추가합니다.

```

/* Record Stop Button */
~

evas_object_show(btn);
ad->btn_recstop = btn;

/* Play Start Button */
btn = elm_button_add(tbl);
elm_object_disabled_set(btn, EINA_TRUE);
elm_object_text_set(btn, "Play Start");
evas_object_smart_callback_add(btn, "clicked", start_player, (void*)ad);
evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.0);
elm_table_pack(tbl, btn, 0, 2, 1, 1);
evas_object_show(btn);
ad->btn_play = btn;

/* Play Stop Button */
btn = elm_button_add(tbl);
elm_object_disabled_set(btn, EINA_TRUE);
elm_object_text_set(btn, "Play Stop");
evas_object_smart_callback_add(btn, "clicked", stop_player, (void*)ad);
evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.0);
elm_table_pack(tbl, btn, 1, 2, 1, 1);
evas_object_show(btn);
ad->btn_playstop = btn;
}

}

// create player
ad->player = create_player();

```

```
/* Show window after base gui is set up */
evas_object_show(ad->win);
```

create_player() 는 Player를 생성하는 함수입니다. 지금부터 만들어 보겠습니다. create_base_gui() 함수 위에 새로운 함수 5개를 추가합니다. AudioPlayer 예제의 소스코드와 거의 유사합니다.

```
// Get player state
static player_state_e get_player_state(player_h player)
{
    player_state_e state;
    player_get_state(player, &state);
    return state;
}

// Stop play
static void
stop_player(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    if( get_player_state(ad->player) == PLAYER_STATE_PLAYING || get_player_state(ad->player) == PLAYER_STATE_PAUSED)
        player_stop(ad->player);

    elm_object_disabled_set(ad->btn_play, EINA_FALSE);
    elm_object_disabled_set(ad->btn_playstop, EINA_TRUE);
    elm_object_disabled_set(ad->btn_rec, EINA_FALSE);
    elm_object_disabled_set(ad->btn_recstop, EINA_TRUE);
}

// Load file to player
```

```

static void
prepare_player(appdata_s* ad)
{
    stop_player(ad, NULL, NULL);
    player_unprepare(ad->player);
    player_set_uri(ad->player, ad->file_path);
    player_prepare(ad->player);
}

// Start play
static void
start_player(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    prepare_player(ad);

    if (get_player_state(ad->player) != PLAYER_STATE_PLAYING)
    {
        player_start(ad->player);

        elm_object_disabled_set(ad->btn_rec, EINA_TRUE);
        elm_object_disabled_set(ad->btn_recstop, EINA_TRUE);
        elm_object_disabled_set(ad->btn_play, EINA_TRUE);
        elm_object_disabled_set(ad->btn_playstop, EINA_FALSE);
    }
}

// Create player
static player_h create_player()
{
    player_h player;

    player_create(&player);
    player_set_completed_cb(player, NULL, player);
}

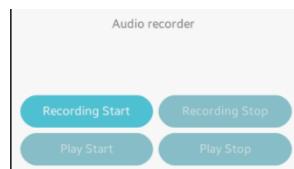
```

```
        return player;  
    }  
}
```

자세한 설명은 `AudioPlayer` 예제를 참조하기 바랍니다.

예제를 실행시켜 봅시다. 데스크탑에서 에뮬레이터로 테스트 할 때는 마이크를 연결해서 테스트 해야 합니다. 폰이나 노트북에서 테스트 할 때는 마이크가 필요하지 않습니다. 다음 과정을 따라서 테스트 해보기 바랍니다.

- Record Start 버튼을 누르고 음성을 녹음합니다.
- Record Stop 버튼을 누르고 녹음을 중지합니다. 20초가 넘어가면 자동으로 중지됩니다.
- Play Start 버튼을 누르면 녹음된 음성이 재생됩니다.
- 중간에 재생을 중지하고 싶으면 Play Stop 버튼을 누릅니다.



5) 관련 API

`int preference_set_int(const char *key, int value)` : 로컬 메모리에 정수값을 저장하는 API. 1번째 파라미터에 키값을 전달하고, 2번째 파라미터에는 데이터를 전달합니다.

`int preference_set_boolean(const char *key, bool value)` : 로컬 메모리에 boolean 값을 저장하는 API. 1번째 파라미터에 키값을 전달하고, 2번째 파라미터에는 데이터를 전달합니다.

int preference_get_boolean(const char *key, bool *value) : 로컬

메모리에서 boolean 형식 데이터를 구하는 API. 1번째 파라미터에 키값을 전달하면 2번째 파라미터에선 결과값이 반환됩니다.

int preference_get_int(const char *key, int *value) : 로컬 메모리에서 정수 형식 데이터를 구하는 API. 1번째 파라미터에 키값을 전달하면 2번째 파라미터에선 결과값이 반환됩니다.

int recorder_create_audiorecorder(recorder_h *recorder) : Recorder를 생성하는 API.

int recorder_set_recording_status_cb(recorder_h recorder, recorder_recording_status_cb callback, void *user_data) : 녹음 상태 변경 이벤트 함수를 지정하는 API.

int recorder_set_recording_limit_reached_cb(recorder_h recorder, recorder_recording_limit_reached_cb callback, void *user_data) : 녹음 최대 시간 도달 이벤트 함수를 지정하는 API.

int recorder_attr_set_audio_channel(recorder_h recorder, int channel_count) : 오디오 채널 개수를 지정하는 API. Mono 인 경우에는 1을 지정하고, Stereo 인 경우에는 2를 지정하면 됩니다.

int recorder_attr_set_audio_device(recorder_h recorder, recorder_audio_device_e device) : 녹음 장비를 지정하는 API.
스마트폰에서는 마이크를 사용하기 때문에
RECORDER_AUDIO_DEVICE_MIC를 지정하면 됩니다.

int recorder_attr_set_time_limit(recorder_h recorder, int second) : 녹음 최대 시간을 지정하는 API. 단위는 초입니다.

`int recorder_set_filename(recorder_h recorder, const char *path)` : 녹음 파일명을 지정하는 API.

`int recorder_set_file_format(recorder_h recorder, recorder_file_format_e format)` : 녹음 포맷을 지정하는 API.

`int recorder_set_audio_encoder(recorder_h recorder, recorder_audio_codec_e codec)` : 코덱을 지정하는 API.

`int recorder_prepare(recorder_h recorder)` : 녹음을 준비하는 API.

`int recorder_start(recorder_h recorder)` : 녹음을 시작하는 API.

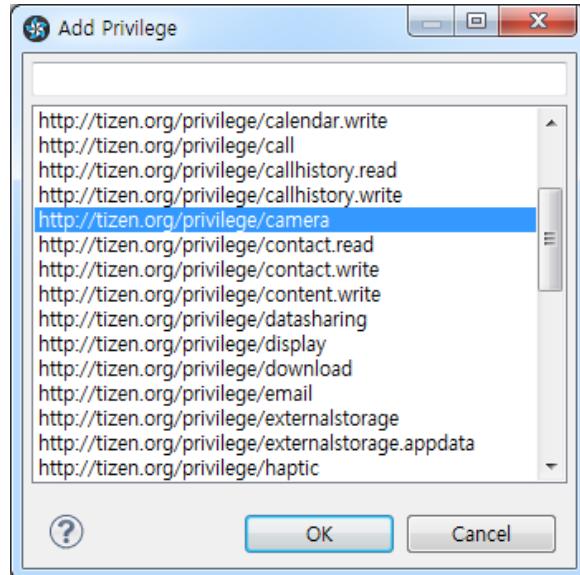
41. 카메라 캡쳐

스마트폰에 내장된 카메라 성능이 일취월장하다 보니 디지털 카메라가 점차 사라지고 있습니다. 실제로 스마트폰 카메라의 성능은 웬만한 디카에 비해 손색이 없을 정도로 발전했고 스마트폰을 고를 때 중요한 요소가 되었습니다. 사진 촬영 이외에도 내장 카메라는 화상통화, 증강현실, 바코드 스캐너 등등 여러가지 용도로 사용됩니다.

이번 시간에는 폰에 장착된 카메라를 이용해서 프리뷰 영상을 보여주고 사진을 촬영해서 파일로 저장하는 방법을 알아보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 CameraEx 으로 지정합니다. 카메라를 사용하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/camera>를 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.

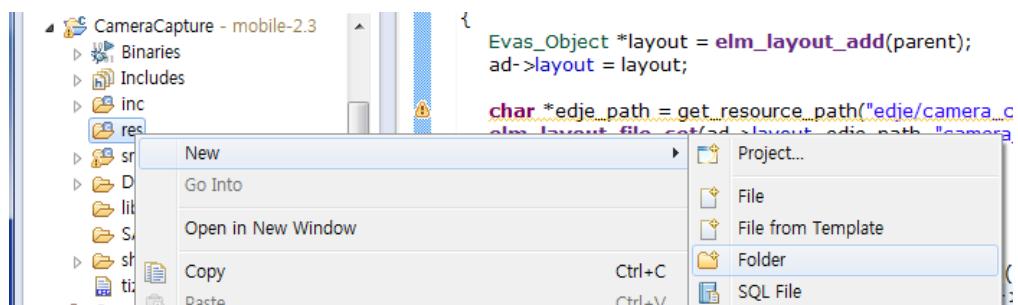


저장하고 아래쪽 템버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

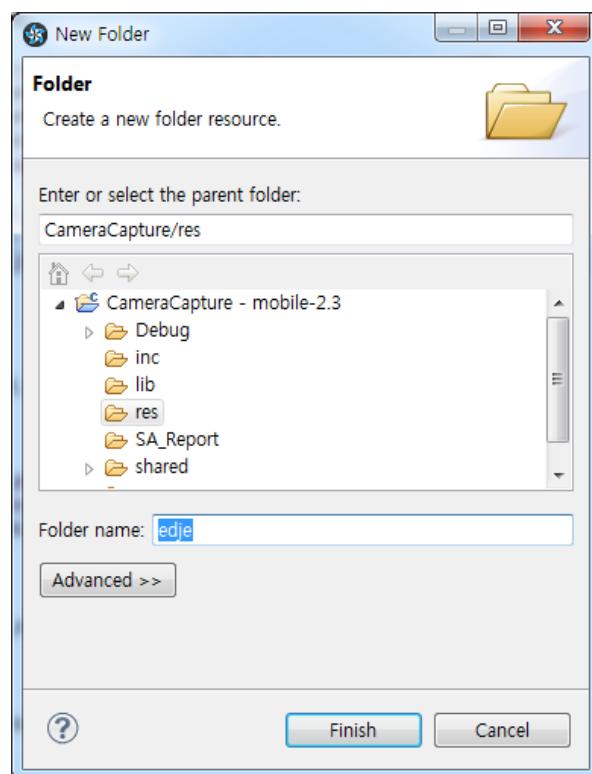
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.cameraex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.cameraex" exec="cameraex" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>cameraex</label>
        <icon>cameraex.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/camera</privilege>
    </privileges>
</manifest>
```

프리뷰 영상 재생할 때는 스크린이 필요합니다. EDJE 파일을 복사해 오겠습니다. 다음 과정을 수행합니다.

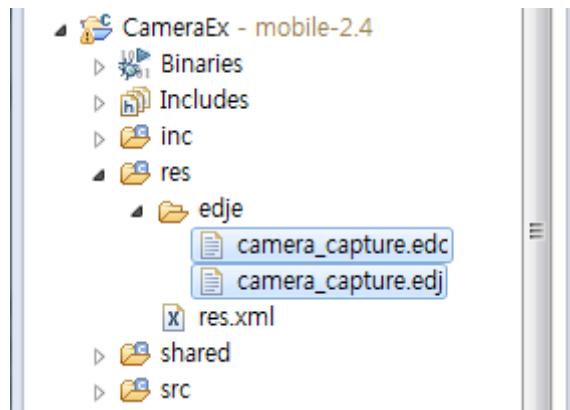
- /res 폴더를 마우스 오른쪽 버튼으로 누르고 단축메뉴 [New > Folder]를 선택합니다.



- 팝업창이 나타나면 Folder name 속성에 edje 라고 입력합니다.



- 새로 생성된 폴더에 부록 /etc/edje 폴더에 있는 camera_capture.edc와 camera_capture.edj 파일을 복사합니다.



2) Camera 프리뷰 영상

src 폴더에 소스파일(~.c)을 열고 화면 위쪽에 라이브러리와 변수를 추가합니다.

```
#include "cameracapture.h"
#include <camera.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
Evas_Object *layout;
Evas_Object *camera_rect;
Evas_Object *image;
Evas_Object *box;

camera_h camera;
```

```
    char *image_path;  
} appdata_s;  
_____
```

프리뷰 영상은 VideoPlayer 예제에서 사용했던 스크린과 유사합니다.
layout 은 프리뷰의 영역에 해당하고, camera_rect 는 프리뷰 영상을
표시하는 Image 객체입니다.

image 객체에는 촬영한 사진을 표시하겠습니다.

camera_h 는 카메라 구조체 입니다.

image_path에는 촬영한 사진을 저장하는 파일 경로입니다.

Camera 객체를 생성하고 프리뷰 영상을 화면에 표시해 보겠습니다.
create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을
추가하는 함수입니다.

```
_____  
static void  
my_box_pack(Evas_Object *box, Evas_Object *child,  
            double h_weight, double v_weight, double h_align, double  
            v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",  
     * "pad_large" and "pad_huge" available as styles in addition to the  
     * "default" frame style */  
    elm_object_style_set(frame, "pad_medium");  
    /* set the input weight/aling on the frame insted of the child */  
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);  
    evas_object_size_hint_align_set(frame, h_align, v_align);  
}
```

```

/* tell the child that is packed into the frame to be able to expand */
evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

/* fill the expanded area (above) as opposed to center in it */
evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
/* actually put the child in the frame and show it */
evas_object_show(child);
elm_object_content_set(frame, child);

}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}

```

그런 다음 `create_base_gui()` 함수에 새로운 코드를 추가합니다. Label 생성 코드는 필요없으니 주석 처리 합니다.

```

static void
create_base_gui(appdata_s *ad)
{
    /* first say that we prefer acceleration via opengl - before we create any windows
*/
    elm_config_accel_preference_set("opengl");

    ~

eext_object_event_callback_add(ad->win, EEXT_CALLBACK_BACK, win_back_cb, ad);

    /* Conformant */
    ad->conform = elm_conformant_add(ad->win);
    elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
    elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);

```

```

    evas_object_size_hint_weight_set(ad->conform,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    elm_win_resize_object_add(ad->win, ad->conform);
    evas_object_show(ad->conform);

    /* Label*/
    /*ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
    evas_object_size_hint_weight_set(ad->label,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    elm_object_content_set(ad->conform, ad->label);*/

    { /* child object - indent to how relationship */
        /* A box to put things in vertically - default mode for box */
        ad->box = elm_box_add(ad->win);
        evas_object_size_hint_weight_set(ad->box,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
        elm_object_content_set(ad->conform, ad->box);
        evas_object_show(ad->box);

        { /* child object - indent to how relationship */
            /* Create preview screen */
            Evas_Object *layout = _main_layout_add(ad, ad->win);
            my_box_pack(ad->box, layout, 0.9, 1.0, -1.0, -1.0);
        }
    }

    _create_camera(ad);
    // Start camera preview
    camera_start_preview(ad->camera);

    /* Show window after base gui is set up */
    evas_object_show(ad->win);
}

```

카메라 프리뷰를 표시하려면 OpenGL 라이브러리를 사용해야 합니다.
그러기 위해서 elm_config_accel_preference_set() 함수에 "opengl"을
전달하면 됩니다.

_main_layout_add() 는 프리뷰 영역에 해당하는 Layout 객체를 생성하는
함수입니다.

_create_camera() 는 Camera 객체를 생성하는 함수입니다.

camera_start_preview() 는 프리뷰를 시작하는 API입니다.

위 함수들을 만들어 보겠습니다. create_base_gui() 함수 위에 새로운
함수 4개를 추가합니다.

```
static inline const char*
get_resource_path(const char *file_path)
{
    static char absolute_path[PATH_MAX] = "";
    static char *res_path_buff = NULL;
    if (res_path_buff == NULL)
        res_path_buff = app_get_resource_path();
    snprintf(absolute_path, sizeof(absolute_path), "%s%s", res_path_buff, file_path);
    return absolute_path;
}

// Create preview screen
static Evas_Object*
_main_layout_add(appdata_s *ad, Evas_Object *parent)
{
    Evas_Object *layout = elm_layout_add(parent);
```

```

ad->layout = layout;

char *edje_path = get_resource_path("edje/camera_capture.edj");
elm_layout_file_set(ad->layout, edje_path, "camera_capture");

Evas *evas = evas_object_evas_get(parent);
ad->camera_rect = evas_object_image_filled_add(evas);
elm_object_part_content_set(layout, "render", ad->camera_rect);

return layout;
}

static void _destroy_camera(appdata_s *ad)
{
    if(ad->camera)
    {
        camera_stop_preview(ad->camera);
        camera_destroy(ad->camera);
        ad->camera = NULL;
    }
}

static void
_create_camera(appdata_s *ad)
{
    if(ad->camera)
        _destroy_camera(ad);

    if(camera_create(CAMERA_DEVICE_CAMERA0, &ad->camera) == CAMERA_ERROR_NONE)
    {
        camera_set_capture_format(ad->camera, CAMERA_PIXEL_FORMAT_JPEG);
        camera_set_display(ad->camera, CAMERA_DISPLAY_TYPE_EVAS, GET_DISPLAY(ad->camera_rect));
        camera_set_display_mode(ad->camera, CAMERA_DISPLAY_MODE_FULL);
    }
}

```

```
    camera_set_display_rotation(ad->camera, CAMERA_ROTATION_270);
    camera_set_display_flip(ad->camera, CAMERA_FLIP_VERTICAL);
}
else
    ad->camera = NULL;
}
```

get_resource_path() 는 /res 폴더에 있는 파일의 절대경로를 반환하는 함수입니다.

_main_layout_add() 는 프리뷰 영역에 해당하는 Layout 객체를 생성하는 함수입니다. 자세한 내용은 VideoPlayer 예제와 동일합니다.

_destroy_camera() 는 Camera 객체를 삭제하는 함수입니다.

camera_stop_preview(camera_h) 는 카메라 프리뷰를 중지하는 API입니다.

camera_destroy(camera_h) 는 Camera 객체를 삭제하는 API입니다.

_create_camera() 는 Camera 객체를 생성하고 프리뷰는 함수입니다.

camera_create(camera_device_e, camera_h *) 는 Camera 객체를 생성하는 API입니다. 1번째 파라미터에 CAMERA_DEVICE_CAMERA0을 전달하면 후면 카메라를 사용합니다. CAMERA_DEVICE_CAMERA1을 전달하면 전면 카메라를 사용합니다. 2번째 파라미터에는 생성된 Camera 객체가 반환됩니다.

camera_set_capture_format(camera_h, camera_pixel_format_e) 는 사진 이미지의 포맷을 지정하는 API입니다. 사진은 용량이 크기 때문에 JPEG 형식으로 지정하는 것이 좋습니다.

`camera_set_display(camera_h, camera_display_type_e, camera_display_h)` 는 Camera 객체에 프리뷰용 스크린을 지정하는 API입니다. Evas 기반의 객체를 사용할 때는 2번째 파라미터에 `CAMERA_DISPLAY_TYPE_EVAS`를 전달하면 됩니다. `GET_DISPLAY()` 함수에 Image 객체를 전달해서 구한 `camera_display_h` 객체를 3번째 파라미터에 전달하면 됩니다.

`camera_set_display_mode(camera_h, camera_display_mode_e)` 는 프리뷰 영상의 확대/축소 옵션을 지정하는 API입니다. 옵션 종류는 다음과 같습니다.

- `CAMERA_DISPLAY_MODE LETTER_BOX` = 0, */*< Letter box */*
- `CAMERA_DISPLAY_MODE_ORIGIN_SIZE`, */*< Origin size */*
- `CAMERA_DISPLAY_MODE_FULL`, */*< Full screen */*
- `CAMERA_DISPLAY_MODE_CROPPED_FULL`, */*< Cropped full screen */*

`camera_set_display_rotation(camera_h, camera_rotation_e)` 는 카메라 회전 각도를 지정하는 API입니다.

`camera_set_display_flip(camera_h, camera_flip_e)` 는 수직/수평 여부를 지정하는 API입니다. 옵션 종류는 다음과 같습니다.

- `CAMERA_FLIP_NONE`, */*< No Flip */*
- `CAMERA_FLIP_HORIZONTAL`, */*< Horizontal flip */*
- `CAMERA_FLIP_VERTICAL`, */*< Vertical flip */*
- `CAMERA_FLIP_BOTH` */* Horizontal and vertical flip */*

예제를 빌드해서 실행시켜 봅시다. 스마트폰 혹은 웹캠이 장착된 PC에서 테스트 해야 합니다. 프리뷰 영상이 화면 위쪽에 표시됩니다.



3) 화면 방향 고정

화면을 수평으로 뉘었을 때 프리뷰의 위치가 이동합니다. 화면 방향이 Portrait에서 Landscape로 변경되었기 때문입니다. 화면 방향을 고정시켜 보겠습니다. `create_base_gui()` 함수 위 부분에 화면 방향 종류를 지정하는 코드가 있습니다. 아래와 같이 수정합니다.

```
if (elm_win_wm_rotation_supported_get(ad->win)) {  
    //int rots[4] = { 0, 90, 180, 270 };  
    int rots[4] = { 0 };  
    //elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots),  
4);  
    elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots), 1);  
}
```

0 은 Portrait Primary를 의미하고 180은 Portrait Secondary를 의미합니다. 90은 Landscape Primary를 의미하고, 270은 Landscape Secondary를 의미합니다. 이제 Portrait Primary 모드만 지원됩니다.

앱이 종료되면 Camera 객체가 자동으로 삭제되도록 하겠습니다. 소스 파일 아래쪽으로 이동해서 `app_terminate()` 함수에 새로운 코드를

추가합니다.

```
static void  
app_terminate(void *data)  
{  
    _destroy_camera(data);  
}
```

app_terminate() 는 앱이 종료될 때 실행되는 콜백 함수입니다. main() 함수에서 콜백 함수를 변경할 수 있습니다.

이제 예제를 다시 실행하고 화면을 회전하면 프리뷰 위치가 변경되지 않습니다.

4) 화면 방향에 따른 프리뷰 회전

이번에는 반대로 화면 방향이 회전할 때 자동으로 카메라를 회전하는 기능을 구현해 보겠습니다. 소스파일 아래쪽에 main() 함수와 ui_app_orient_changed() 함수의 코드를 수정합니다.

```
static void  
ui_app_orient_changed(app_event_info_h event_info, void *user_data)  
{  
    appdata_s *ad = user_data;  
    app_device_orientation_e screen_rot = 0;  
    camera_rotation_e camera_rot = CAMERA_ROTATION_270;  
    bool horizontal_box = false;  
  
    /* Properly handle rotation */
```

```

app_event_get_device_orientation(event_info, &screen_rot);
switch (screen_rot)
{
    case APP_DEVICE_ORIENTATION_0:
        camera_rot = CAMERA_ROTATION_270;
        break;
    case APP_DEVICE_ORIENTATION_90:
        camera_rot = CAMERA_ROTATION_180;
        horizontal_box = true;
        break;
    case APP_DEVICE_ORIENTATION_180:
        camera_rot = CAMERA_ROTATION_90;
        break;
    case APP_DEVICE_ORIENTATION_270:
        camera_rot = CAMERA_ROTATION_NONE;
        horizontal_box = true;
        break;
}
/* Set camera preview rotation */
camera_set_display_rotation(ad->camera, camera_rot);
/* Make sure to rotate the window itself */
elm_win_rotation_with_resize_set(ad->win, screen_rot);
/* Relayout elements in the window by choosing horizontal vs. vertical box
*/
elm_box_horizontal_set(ad->box, horizontal_box);
}

int
main(int argc, char *argv[])
{
    appdata_s ad = {0,};
    int ret = 0;

```

```

ui_app_lifecycle_callback_s event_callback = {0,};
app_event_handler_h handlers[5] = {NULL, };

```

```

event_callback.create = app_create;
event_callback.terminate = app_terminate;
event_callback.pause = app_pause;
event_callback.resume = app_resume;
event_callback.app_control = app_control;

ui_app_add_event_handler(&handlers[APP_EVENT_DEVICE_ORIENTATION_C
HANGED], APP_EVENT_DEVICE_ORIENTATION_CHANGED, ui_app_orient_changed,
&ad);

ui_app_add_event_handler(&handlers[APP_EVENT_LANGUAGE_CHANGED],
APP_EVENT_LANGUAGE_CHANGED, ui_app_lang_changed, &ad);

ret = ui_app_main(argc, argv, &event_callback, &ad);
if (ret != APP_ERROR_NONE) {
    dlog_print(DLOG_ERROR, LOG_TAG, "app_main() is failed. err = %d",
ret);
}

return ret;
}

```

main() 함수에서 화면 방향 변경 이벤트 함수를 ui_app_orient_changed() 으로 지정합니다. 만약 이 코드가 없다면 추가하면 됩니다.

그런 다음 ui_app_orient_changed() 함수에 새로운 코드를 추가합니다.

app_event_get_device_orientation() 는 이벤트 객체에서 화면 방향을 구하는 API입니다. APP_DEVICE_ORIENTATION_0 또는 APP_DEVICE_ORIENTATION_180 이면 Portrait 모드이고, APP_DEVICE_ORIENTATION_90 또는 APP_DEVICE_ORIENTATION_270 이면 Landscape 모드입니다.

camera_set_display_rotation() 은 카메라를 회전하는 API 입니다.

elm_win_rotation_with_resize_set() 은 화면 방향에 따라서 Window의 크기를 재조정하는 API 입니다.

5) 카메라 캡쳐

Button을 누르면 프리뷰 영상을 캡쳐해서 이미지 파일로 저장하는 기능을 구현해 보겠습니다. create_base_gui() 함수에 Button 생성 코드를 추가합니다.

```
{ /* child object - indent to how relationship */
    /* Create preview screen */
    Evas_Object *layout = _main_layout_add(ad, ad->win);
    my_box_pack(ad->box, layout, 0.9, 1.0, -1.0, -1.0);

    /* Capture button */
    Evas_Object *btn = elm_button_add(ad->win);
    elm_object_text_set(btn, "#");
    evas_object_smart_callback_add(btn, "clicked", btn_capture_cb, ad);
    my_box_pack(ad->box, btn, 0.1, 0.0, -1.0, 0.5);
}
```

그런 다음 create_base_gui() 함수 위에 새로운 함수 5개를 추가합니다.

```
static inline char*
gen_data_path(const char *file_name)
{
```

```

static char *absolute_path = NULL;
char result[PATH_MAX] = "";
if (absolute_path == NULL)
    absolute_path = app_get_data_path();
snprintf(result, sizeof(result), "%s/%s", absolute_path, file_name);
return strdup(result);
}

// Save image data to file
static char*
_save_file(appdata_s *ad, camera_image_data_s *image)
{
    char buf[PATH_MAX] = "";
    snprintf(buf, PATH_MAX, "camera_capture.jpg");
    char *file_name = gen_data_path(buf);

    FILE *f = fopen(file_name, "w");

    if(f)
    {
        fwrite(image->data, image->size, 1, f);
        fclose(f);
    }
    else
    {
        free(file_name);
        file_name = NULL;
    }
    return file_name;
}

static void
_on_camera_capture_cb(camera_image_data_s *image, camera_image_data_s *postview,
camera_image_data_s *thumbnail, void *user_data)
{

```

```

    appdata_s *ad = user_data;
    free(ad->image_path);
}

static void _on_camera_capture_completed_cb(void *user_data)
{
    appdata_s *ad = user_data;
    camera_start_preview(ad->camera);
}

static void
btn_capture_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = (appdata_s *)data;
    camera_start_capture(ad->camera,
                         _on_camera_capture_cb,
                         _on_camera_capture_completed_cb, ad);
}

```

`gen_data_path()` 는 앱 내부 `/data` 폴더의 파일 경로를 반환하는 함수입니다. `/res` 폴더에는 파일을 생성하거나 수정할 수 없습니다. 그래서 `/data` 폴더 혹은 공용 폴더에 저장해야 합니다.

`app_get_data_path()` 는 `/data` 폴더의 절대 경로를 반환하는 API입니다.

`_save_file()` 는 이미지 데이터를 파일에 저장하는 함수입니다.

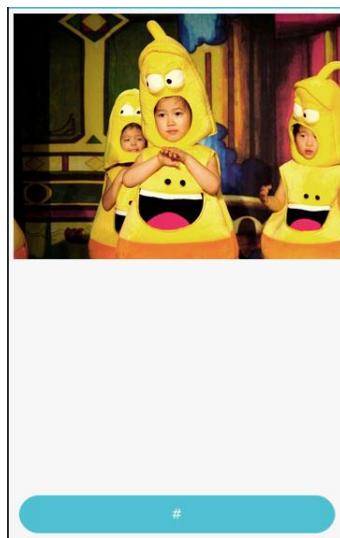
`_on_camera_capture_cb()` 는 카메라 캡쳐 데이터를 수신하는 콜백 함수입니다. 이 함수에서 데이터를 저장하면 됩니다.

`_on_camera_capture_completed_cb()` 는 카메라 캡쳐 완료 이벤트 콜백 함수입니다. 캡쳐 후에는 프리뷰가 종료되기 때문에 이 함수에서 프리뷰를 재시작 해주어야 합니다.

`btn_capture_cb()` 는 Button 콜백 함수입니다.

`camera_start_capture(camera_h, camera_capturing_cb,`
`camera_capture_completed_cb, void *)` 는 카메라 캡쳐를 시작하는
API입니다. 1번째 파라미터는 Camera 객체, 2번째는 데이터 수신 함수,
3번째는 캡쳐 완료 이벤트 콜백 함수, 4번째는 사용자 데이터 입니다.

예제를 다시 실행하고 Button을 눌러봅시다. 카메라 소리가 들리면
정상적으로 캡쳐된 것입니다. 저장된 파일을 보여주는 기능은 아직
구현되지 않았습니다.



5) 저장된 이미지 파일을 화면에 표시

저장된 이미지 파일을 Image 객체에 표시하는 기능을 구현해
보겠습니다. `_main_layout_add()` 함수에 Image 객체를 생성하는 코드를
추가합니다.

```

// Create preview screen
static Evas_Object*
_main_layout_add(appdata_s *ad, Evas_Object *parent)
{
    ~

    Evas *evas = evas_object_evas_get(parent);
    ad->camera_rect = evas_object_image_filled_add(evas);
    elm_object_part_content_set(layout, "render", ad->camera_rect);

ad->image = elm_image_add(parent);
elm_object_part_content_set(layout, "gallery", ad->image);

    return layout;
}

```

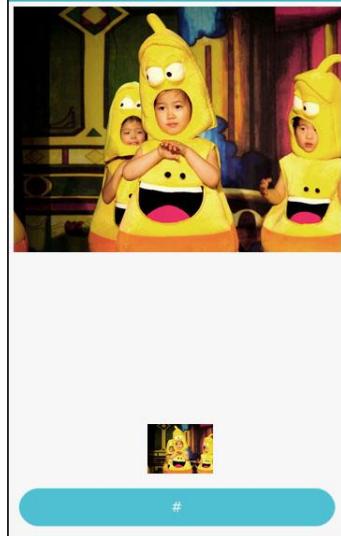
카메라 캡쳐가 완료되면 이미지 파일을 Image 객체에 로딩하면 됩니다.
`_on_camera_capture_completed_cb()` 함수에 새로운 코드를 추가합니다.

```

static void
_on_camera_capture_cb(camera_image_data_s *image, camera_image_data_s *postview,
camera_image_data_s *thumbnail, void *user_data)
{
    appdata_s *ad = user_data;
    free(ad->image_path);
ad->image_path = _save_file(ad, image);
}

```

예제를 다시 실행하고 Button 누르면 캡쳐 이미지가 bg 위젯에 표시됩니다.



6) 관련 API

int camera_start_preview(camera_h camera) : 프리뷰를 시작하는 API.

int camera_stop_preview(camera_h camera) : 카메라 프리뷰를 중지하는 API.

int camera_destroy(camera_h camera) : Camera 객체를 삭제하는 API.

int camera_create(camera_device_e device, camera_h *camera) : Camera 객체를 생성하는 API. 1번째 파라미터에 CAMERA_DEVICE_CAMERA0을 전달하면 후면 카메라를 사용합니다. CAMERA_DEVICE_CAMERA1을 전달하면 전면 카메라를 사용합니다. 2번째 파라미터에는 생성된 Camera 객체가 반환됩니다.

int camera_set_capture_format(camera_h camera, camera_pixel_format_e format) : 사진 이미지의 포맷을 지정하는 API.

CAMERA_PIXEL_FORMAT_JPEG 으로 지정하면 JPEG 형식으로 이미지가

만들어 집니다.

int camera_set_display(camera_h camera, camera_display_type_e type, camera_display_h display) : Camera 객체에 프리뷰용 스크린을 지정하는 API. Evas 기반의 객체를 사용할 때는 2번째 파라미터에 CAMERA_DISPLAY_TYPE_EVAS를 전달하면 됩니다. GET_DISPLAY() 함수에 Image 객체를 전달해서 구한 camera_display_h 객체를 3번째 파라미터에 전달하면 됩니다.

int camera_set_display_mode(camera_h camera, camera_display_mode_e mode) : 프리뷰 영상의 확대/축소 옵션을 지정하는 API. 옵션 종류는 다음과 같습니다.

- CAMERA_DISPLAY_MODE_LETTER_BOX = 0, /**< Letter box */
- CAMERA_DISPLAY_MODE_ORIGIN_SIZE, /**< Origin size */
- CAMERA_DISPLAY_MODE_FULL, /**< Full screen */
- CAMERA_DISPLAY_MODE_CROPPED_FULL, /**< Cropped full screen */

int camera_set_display_rotation(camera_h camera, camera_rotation_e rotation) : 카메라 회전 각도를 지정하는 API.

int camera_set_display_flip(camera_h camera, camera_flip_e flip) : 수직/수평 여부를 지정하는 API. 옵션 종류는 다음과 같습니다.

- CAMERA_FLIP_NONE, /**< No Flip */
- CAMERA_FLIP_HORIZONTAL, /**< Horizontal flip */
- CAMERA_FLIP_VERTICAL, /**< Vertical flip */
- CAMERA_FLIP_BOTH /**< Horizontal and vertical flip */

char *app_get_data_path(void) : /data 폴더의 절대 경로를 반환하는 API.

```
int camera_start_capture(camera_h camera, camera_capturing_cb  
capturing_cb, camera_capture_completed_cb completed_cb, void  
*user_data) : 카메라 캡쳐를 시작하는 API. / 파라미터 : Camera 객체,  
데이터 수신 함수, 캡쳐 완료 이벤트 콜백 함수, 사용자 데이터.
```

42. 시스템 정보

앱을 개발할 때 도움말 화면에 시스템 정보를 표시해주는 경우가 많습니다. 카메라를 사용하는 앱을 개발할 때는 후면 카메라 혹은 전면 카메라가 존재하는지 여부를 확인해야 합니다. 다양한 해상도의 장비를 호환하려면 모니터 픽셀 개수를 구해야 합니다. 이번 예제에서는 시스템 정보를 구하는 방법을 알아보겠습니다.

1) 후면 카메라 존재 여부

새로운 소스 프로젝트를 생성하고 Project name을 SystemInfo 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 소스파일 위쪽에 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "systeminfo.h"  
#include <system_info.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label1;  
    Evas_Object *label2;  
    Evas_Object *label3;  
    Evas_Object *label4;  
    Evas_Object *label5;  
    Evas_Object *label6;  
} appdata_s;
```

총 6개의 Label 위젯 변수를 선언하였습니다. 1번째 Label에는 후면 카메라 존재 여부, 2번째에는 전면 카메라 존재 여부, 3번째에는 전화통화 가능 여부, 4번째에는 모니터 수평 픽셀 개수, 5번째에는 모니터 수직 픽셀 개수, 6번째에는 플랫폼 버전을 표시해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수 3개를 생성합니다.

```
static void  
my_table_pack(Evas_Object *table, Evas_Object *child, int col, int row, int spanx, int  
spany,  
    bool h_expand, bool v_expand, double h_align, double v_align)  
{  
    /* Create a frame around the child, for padding */  
    Evas_Object *frame = elm_frame_add(table);  
    elm_object_style_set(frame, "pad_small");  
  
    evas_object_size_hint_weight_set(frame, h_expand ? EVAS_HINT_EXPAND : 0,  
v_expand ? EVAS_HINT_EXPAND : 0);  
    evas_object_size_hint_align_set(frame, h_align, v_align);  
  
    /* place child in its box */  
    {  
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);  
        elm_object_content_set(frame, child);  
        evas_object_show(child);  
    }  
  
    elm_table_pack(table, frame, col, row, spanx, spany);  
    evas_object_show(frame);  
}  
  
static Evas_Object *
```

```

my_button_add(Evas_Object *parent, const char *text, Evas_Smart_Cb cb, void *cb_data)
{
    Evas_Object *btn;

    btn = elm_button_add(parent);
    elm_object_text_set(btn, text);
    evas_object_smart_callback_add(btn, "clicked", cb, cb_data);

    return btn;
}

static Evas_Object *
my_label_add(Evas_Object *parent, const char *text)
{
    Evas_Object *btn;

    btn = elm_label_add(parent);
    elm_object_text_set(btn, text);

    return btn;
}

```

my_table_pack() 은 Table 컨테이너에 위젯을 추가하는 함수입니다.

my_button_add() 는 Button 위젯을 생성하는 함수입니다.

my_label_add() 는 Label 위젯을 생성하는 함수입니다.

create_base_gui() 함수 안에 새로운 코드를 추가합니다. 1개의 Frame, Table, Button 과 12개의 Label 위젯을 생성하는 코드입니다.

/* Conformant */

```

ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

Evas_Object *tbl, *btn, *frame, *o;

/* Frame */
frame = elm_frame_add(ad->win);
elm_object_style_set(frame, "pad_medium");
elm_object_content_set(ad->conform, frame);
evas_object_show(frame);

/* Container: standard table */
tbl = elm_table_add(ad->win);
/* Make this table homogeneous for nicer, fixed layout */
elm_table_homogeneous_set(tbl, EINA_TRUE);
elm_object_content_set(frame, tbl);
evas_object_show(tbl);

{

/* Button */
btn = my_button_add(tbl, "Load System Info", btn_clicked_cb, ad);
my_table_pack(tbl, btn, 0, 0, 2, 1, EVAS_HINT_EXPAND, 0,
EVAS_HINT_FILL, EVAS_HINT_FILL);

o = my_label_add(tbl, "Back Camera:");
my_table_pack(tbl, o, 0, 1, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

ad->label1 = my_label_add(tbl, "");

```

```

        my_table_pack(tbl, ad->label1, 1, 1, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

        o = my_label_add(tbl, "Front Camera:");
        my_table_pack(tbl, o, 0, 2, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label2 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label2, 1, 2, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

        o = my_label_add(tbl, "Telephony:");
        my_table_pack(tbl, o, 0, 3, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label3 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label3, 1, 3, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

        o = my_label_add(tbl, "Screen Width:");
        my_table_pack(tbl, o, 0, 4, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label4 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label4, 1, 4, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

        o = my_label_add(tbl, "Screen Height:");
        my_table_pack(tbl, o, 0, 5, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label5 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label5, 1, 5, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

```

```

    o = my_label_add(tbl, "Platform Version:");
    my_table_pack(tbl, o, 0, 6, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

    ad->label6 = my_label_add(tbl, "");
    my_table_pack(tbl, ad->label6, 1, 6, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Button 콜백 함수를 생성하겠습니다. create_base_gui() 함수 위에 새로운 코드를 추가합니다.

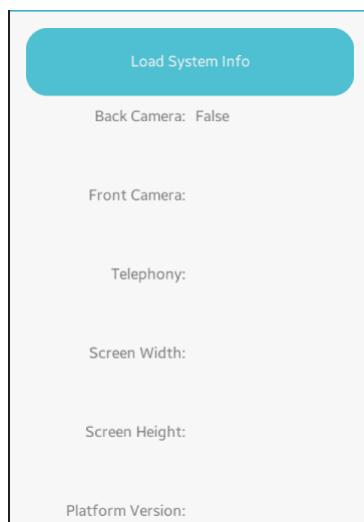
```

static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char buf[100];
    char *sValue = NULL;
    bool bValue = false;
    int nValue = 0;
    int ret;

    ret = system_info_get_platform_bool("http://tizen.org/feature/camera.back",
&bValue);
    if (ret == SYSTEM_INFO_ERROR_NONE)
    {
        elm_object_text_set(ad->label1, bValue ? "True" : "False");
    }
}
```

`system_info_get_platform_bool(char *, bool *)` 는 시스템 정보를 구하는 API 입니다. 반환되는 데이터 형식은 boolean입니다. 1번째 파라미터는 키값이고, "http://tizen.org/feature/camera.back"를 전달하면 후면 카메라 존재 여부를 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 1번째 Label에 텍스트가 변경됩니다. 에뮬레이터에서 실행하면 False가 표시되고, 단말에서 실행하면 True가 표시됩니다.



2) 전면 카메라 존재 여부

이번에는 전면 카메라가 존재하는지 여부를 화면에 표시해 보겠습니다. `btn_clicked_cb()` 함수 끝부분에 새로운 코드를 추가합니다.

```
ret      =      system_info_get_platform_bool("http://tizen.org/feature/camera.back",
&bValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
```

```

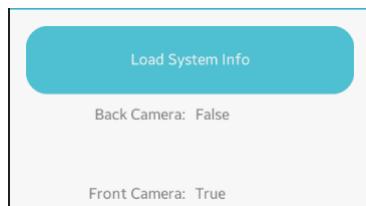
    elm_object_text_set(ad->label1, bValue ? "True" : "False");
}

ret = system_info_get_platform_bool("http://tizen.org/feature/camera.front",
&bValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    elm_object_text_set(ad->label2, bValue ? "True" : "False");
}
}

```

system_info_get_platform_bool() 함수의 1번째 파라미터에 "http://tizen.org/feature/camera.front"를 전달하면 전면 카메라 존재 여부를 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 2번째 Label에 True 가 표시됩니다.



3) 전화 기능 존재 여부

이번에는 전화 기능이 존재하는지 여부를 확인해 보겠습니다. btn_clicked_cb() 함수 끝부분에 새로운 코드를 추가합니다.

```
ret = system_info_get_platform_bool("http://tizen.org/feature/camera.front",
```

```

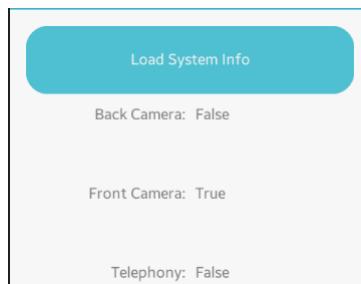
&bValue);
    if (ret == SYSTEM_INFO_ERROR_NONE)
    {
        elm_object_text_set(ad->label2, bValue ? "True" : "False");
    }

ret = system_info_get_platform_bool("http://tizen.org/feature/network.telephony",
&bValue);
    if (ret == SYSTEM_INFO_ERROR_NONE)
    {
        elm_object_text_set(ad->label3, bValue ? "True" : "False");
    }
}

```

system_info_get_platform_bool() 함수의 1번째 파라미터에 "http://tizen.org/feature/network.telephony"를 전달하면 전화 기능 존재 여부를 반환합니다. 여기서 true 값이 반환된다고 해서 전화 통화나 네트워크를 사용할 수 있다는 의미는 아닙니다. 이것은 어디까지나 하드웨어 통신 기능이 장착되어 있다는 의미입니다. USIM 칩이 없거나 환경설정에서 네트워크 기능을 비활성화 해놓은 상태라면 통신을 사용할 수 없습니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 3번째 Label에 True가 표시됩니다.



4) 모니터 픽셀 개수

이번에는 모니터의 픽셀 개수를 구해 보겠습니다. `btn_clicked_cb()` 함수 끝부분에 새로운 코드를 추가합니다.

```
ret = system_info_get_platform_bool("http://tizen.org/feature/network.telephony",
&bValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    elm_object_text_set(ad->label3, bValue ? "True" : "False");
}

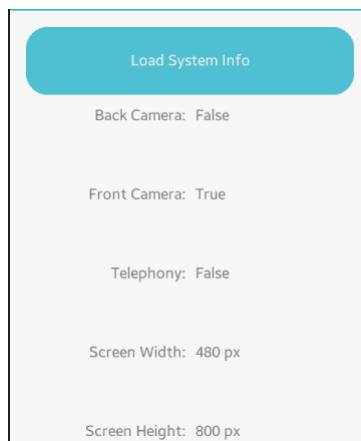
ret = system_info_get_platform_int("tizen.org/feature/screen.width", &nValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "%d px", nValue);
    elm_object_text_set(ad->label4, buf);
}

ret = system_info_get_platform_int("tizen.org/feature/screen.height", &nValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "%d px", nValue);
    elm_object_text_set(ad->label5, buf);
}
```

`system_info_get_platform_int(char *, int *)` 는 시스템 정보를 구하는 API입니다. 반환되는 데이터 형식은 정수형 입니다. 1번째 파라미터는 키값이고, "http://tizen.org/feature/screen.width"를 전달하면 모니터 수평 픽셀 개수를 반환합니다. "tizen.org/feature/screen.height"를 전달하면

모니터 수직 픽셀 개수를 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 4번째 Label과 5번째 Label에 숫자가 표시됩니다.



5) 플랫폼 버전

이번에는 플랫폼 버전 정보를 구해 보겠습니다. `btn_clicked_cb()` 함수 끝부분에 새로운 코드를 추가합니다.

```
ret = system_info_get_platform_int("tizen.org/feature/screen.height", &nValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "%d px", nValue);
    elm_object_text_set(ad->label5, buf);
}

ret =  

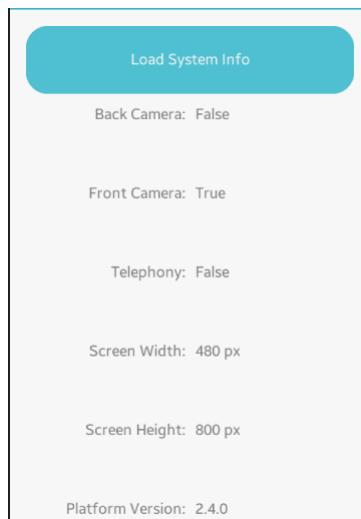
system_info_get_platform_string("http://tizen.org/feature/platform.version",
&sValue);  

if (ret == SYSTEM_INFO_ERROR_NONE)
```

```
{  
    elm_object_text_set(ad->label6, sValue);  
    free(sValue);  
}  
}  
_____
```

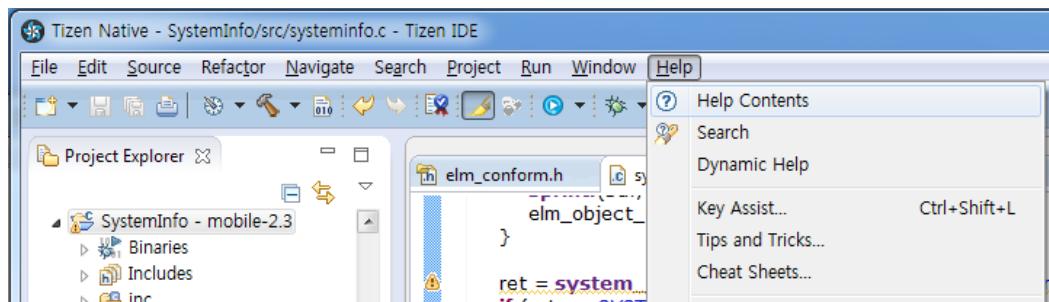
system_info_get_platform_string(char *, char **) 는 시스템 정보를 구하는 API입니다. 반환되는 데이터 형식은 문자열 입니다. 1번째 파라미터는 키값이고, "http://tizen.org/feature/platform.version"를 전달하면 플랫폼 버전을 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 6번째 Label에 플랫폼 버전이 표시됩니다.



6) 관련 API

시스템 정보에는 어떤 종류가 있고 키값과 반환형식은 어떻게 되는지 Help Contents에서 목록을 확인해 보겠습니다. 메인메뉴 [Help > Help Contents]를 선택합니다.



Help Contents가 실행되면 왼쪽 트리 목록에서 [Tizen Mobile Native App Programming > Programming Guide > System > System Information]을 선택합니다. 오른쪽 화면에 키와 반환 형식, 그리고 설명이 표시됩니다.

A screenshot of the Tizen Help Contents window. The left sidebar shows a tree view of topics under 'Tizen Mobile Native App Programming'. The 'System Information' topic is selected. The main pane displays three tables: 'Table: Camera feature keys', 'Table: FM radio feature keys', and 'Table: Graphics feature keys'. Each table lists a key, its type (bool), and a description. The 'Camera feature keys' table includes rows for camera, camera.back, camera.back.flash, camera.front, and camera.front.flash. The 'FM radio feature keys' table includes a row for fmradio. The 'Graphics feature keys' table is empty.

`int system_info_get_platform_bool(const char *key, bool *value)` : 시스템 정보를 구하는 API. 반환되는 데이터 형식은 boolean.

`int system_info_get_platform_int(const char *key, int *value)` : 시스템 정보를 구하는 API. 반환되는 데이터 형식은 정수형.

`int system_info_get_platform_string(const char *key, char **value)` : 시스템 정보를 구하는 API. 반환되는 데이터 형식은 문자열.

43. 시스템 환경설정 정보

다국어를 지원하려면 사용자의 언어 설정을 확인해야 합니다. 새로운 메시지가 도착했을 때 무음 모드 상태라면 진동으로 알림을 대신해야 합니다. 이번 예제에서는 시스템 환경설정 정보를 구하는 방법을 알아보겠습니다.

1) 언어 설정 구하기

새로운 소스 프로젝트를 생성하고 Project name을 SystemSetting 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 소스파일 위쪽에 변수를 추가합니다.

```
#include "systemsetting.h"

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
Evas_Object *label1;
Evas_Object *label2;
Evas_Object *label3;
Evas_Object *label4;
} appdata_s;
```

총 4개의 Label 위젯 변수를 선언하였습니다. 1번째 Label에는 언어 설정, 2번째에는 무음 모드 여부, 3번째에는 TimeZone, 4번째에는 장비명을 표시해 보겠습니다.

`create_base_gui()` 함수 위에 새로운 함수 3개를 생성합니다.

```
static void
my_table_pack(Evas_Object *table, Evas_Object *child, int col, int row, int spanx, int
spany,
              bool h_expand, bool v_expand, double h_align, double v_align)
{
    /* Create a frame around the child, for padding */
    Evas_Object *frame = elm_frame_add(table);
    elm_object_style_set(frame, "pad_small");

    evas_object_size_hint_weight_set(frame, h_expand ? EVAS_HINT_EXPAND : 0,
v_expand ? EVAS_HINT_EXPAND : 0);
    evas_object_size_hint_align_set(frame, h_align, v_align);

    /* place child in its box */
    {
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        elm_object_content_set(frame, child);
        evas_object_show(child);
    }

    elm_table_pack(table, frame, col, row, spanx, spany);
    evas_object_show(frame);
}

static Evas_Object *
my_button_add(Evas_Object *parent, const char *text, Evas_Smart_Cb cb, void *cb_data)
{
    Evas_Object *btn;

    btn = elm_button_add(parent);
```

```

elm_object_text_set(btn, text);
evas_object_smart_callback_add(btn, "clicked", cb, cb_data);

return btn;
}

static Evas_Object *
my_label_add(Evas_Object *parent, const char *text)
{
    Evas_Object *btn;

    btn = elm_label_add(parent);
    elm_object_text_set(btn, text);

    return btn;
}

```

my_table_pack() 은 Table 컨테이너에 위젯을 추가하는 함수입니다.

my_button_add() 는 Button 위젯을 생성하는 함수입니다.

my_label_add() 는 Label 위젯을 생성하는 함수입니다.

create_base_gui() 함수 안에 새로운 코드를 추가합니다. 1개의 Frame, Table, Button 위젯과 8개의 Label 위젯을 생성하는 코드입니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

```

```

elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
    Evas_Object *tbl, *btn, *frame, *o;

    /* Frame */
    frame = elm_frame_add(ad->win);
    elm_object_style_set(frame, "pad_medium");
    elm_object_content_set(ad->conform, frame);
    evas_object_show(frame);

    /* Container: standard table */
    tbl = elm_table_add(ad->win);
    /* Make this table homogeneous for nicer, fixed layout */
    elm_table_homogeneous_set(tbl, EINA_TRUE);
    elm_object_content_set(frame, tbl);
    evas_object_show(tbl);

    {
        /* Button */
        btn = my_button_add(tbl, "Load System Settings", btn_clicked_cb, ad);
        my_table_pack(tbl, btn, 0, 0, 2, 1, EVAS_HINT_EXPAND, 0,
EVAS_HINT_FILL, EVAS_HINT_FILL);

        /* Fields */
        o = my_label_add(tbl, "Language:");
        my_table_pack(tbl, o, 0, 1, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label1 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label1, 1, 1, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

        o = my_label_add(tbl, "Silent mode:");
    }
}

```

```

        my_table_pack(tbl, o, 0, 2, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label2 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label2, 1, 2, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

        o = my_label_add(tbl, "Time zone:");
        my_table_pack(tbl, o, 0, 3, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label3 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label3, 1, 3, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);

        o = my_label_add(tbl, "Device name:");
        my_table_pack(tbl, o, 0, 4, 1, 1, EVAS_HINT_EXPAND, 0, 1.0,
EVAS_HINT_FILL);

        ad->label4 = my_label_add(tbl, "");
        my_table_pack(tbl, ad->label4, 1, 4, 1, 1, EVAS_HINT_EXPAND, 0, 0.0,
EVAS_HINT_FILL);
    }

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Button 콜백 함수를 생성하겠습니다. `create_base_gui()` 함수 위에 새로운 코드를 추가합니다.

```

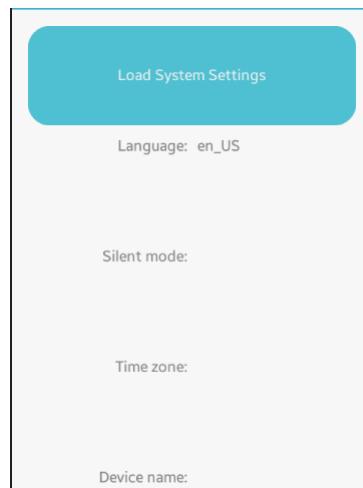
static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)

```

```
{  
    appdata_s *ad = data;  
    char buf[100];  
    char *sValue = NULL;  
    bool bValue;  
  
    system_settings_get_value_string(SYSTEM_SETTINGS_KEY_LOCALE_LANGUAGE,  
    &sValue);  
    elm_object_text_set(ad->label1, sValue);  
    free(sValue);  
}
```

system_settings_get_value_string(system_settings_key_e, char **) 는 시스템 환경설정 정보를 구하는 API 입니다. 반환되는 데이터 형식은 문자열 입니다. 1번째 파라미터는 키값이고, SYSTEM_SETTINGS_KEY_LOCALE_LANGUAGE를 전달하면 사용자가 지정한 언어 종류를 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 1번째 Label에 언어 종류가 표시됩니다.



2) 무음 모드 구하기

이번에는 무음 모드 상태인지 여부를 화면에 표시해 보겠습니다.

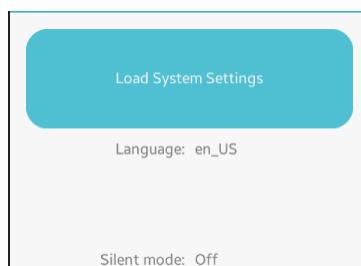
btn_clicked_cb() 함수 끝부분에 새로운 코드를 추가합니다.

```
    system_settings_get_value_string(SYSTEM_SETTINGS_KEY_LOCALE_LANGUAGE,
&sValue);
    elm_object_text_set(ad->label1, sValue);
    free(sValue);

    system_settings_get_value_bool(SYSTEM_SETTINGS_KEY_SOUND_SILENT_MODE,
&bValue);
    elm_object_text_set(ad->label2, bValue ? "On" : "Off");
}
```

system_settings_get_value_bool(system_settings_key_e, bool *) 는 시스템 환경설정 정보를 구하는 API 입니다. 반환되는 데이터 형식은 bool 입니다. 1번째 파라미터는 키값이고, SYSTEM_SETTINGS_KEY_SOUND_SILENT_MODE를 전달하면 무음 모드 여부를 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 2번째 Label에 무음 모드 여부가 표시됩니다.



3) TimeZone 구하기

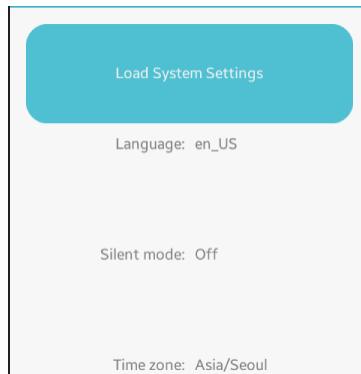
이번에는 TimeZone을 구해해 보겠습니다. `btn_clicked_cb()` 함수 끝부분에 새로운 코드를 추가합니다.

```
    system_settings_get_value_bool(SYSTEM_SETTINGS_KEY_SOUND_SILENT_MODE,
&bValue);
    elm_object_text_set(ad->label2, bValue ? "On" : "Off");

    system_settings_get_value_string(SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE,
&sValue);
    elm_object_text_set(ad->label3, sValue);
    free(sValue);
}
```

`system_settings_get_value_string()` 함수의 1번째 파라미터에 `SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE`을 전달하면 TimeZone을 문자열로 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 3번째 Label에 사용자가 설정한 TimeZone 이 표시됩니다.



4) 장비명 구하기

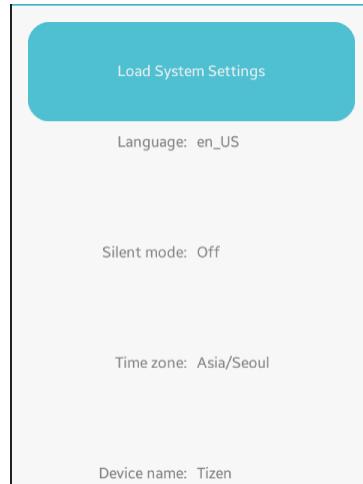
이번에는 장비명을 구해 보겠습니다. `btn_clicked_cb()` 함수 끝부분에 새로운 코드를 추가합니다.

```
    system_settings_get_value_string(SYSTEM_SETTINGS_KEY_LOCALE_TIMEZONE,
&sValue);
    elm_object_text_set(ad->label3, sValue);
    free(sValue);

    system_settings_get_value_string(SYSTEM_SETTINGS_KEY_DEVICE_NAME,
&sValue);
    elm_object_text_set(ad->label4, sValue);
    free(sValue);
}
```

`system_settings_get_value_string()` 함수의 1번째 파라미터에 `SYSTEM_SETTINGS_KEY_DEVICE_NAME`을 전달하면 장비명을 문자열로 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 4번째 Label에 장비명이 표시됩니다.



5) 관련 API

시스템 환경설정 정보에는 어떤 종류가 있고 키값과 반환형식은 어떻게 되는지 목록을 확인해 보겠습니다. 웹 브라우저를 실행하고 아래 주소로 이동하면 System Information 관련 키값 리스트를 확인하실수 있습니다.

<https://developer.tizen.org/documentation/guides/native-application/system/system-information>

KEY	TYPE	DESCRIPTION
<code>http://tizen.org/feature/camera</code>	<code>bool</code>	The platform returns <code>true</code> for this key, if the device provides any kind of a camera.
<code>http://tizen.org/feature/camera.back</code>	<code>bool</code>	The platform returns <code>true</code> for this key and the <code>http://tizen.org/feature/camera</code> key, if the device provides a back-facing camera.
<code>http://tizen.org/feature/camera.back.flash</code>	<code>bool</code>	The platform returns <code>true</code> for this key and the <code>http://tizen.org/feature/camera.back</code> key, if the device provides a back-facing camera with a flash.
<code>http://tizen.org/feature/camera.front</code>	<code>bool</code>	The platform returns <code>true</code> for this key and the <code>http://tizen.org/feature/camera</code> key, if the device provides a front-facing camera.

`int system_settings_get_value_int(system_settings_key_e key, int *value)` : 시스템 환경설정 정보를 구하는 API. 반환되는 데이터 형식은 정수형.

`int system_settings_get_value_bool(system_settings_key_e key, bool *value)` : 시스템 환경설정 정보를 구하는 API. 반환되는 데이터 형식은 bool형.

`int system_settings_get_value_string(system_settings_key_e key, char **value)` : 시스템 환경설정 정보를 구하는 API. 반환되는 데이터 형식은 문자열.

44. 배터리 상태 정보

동영상 재생기 앱을 만들때 배터리 잔량이 15% 이하로 내려가면 동영상 재생을 중지해야 합니다. 중요한 전화를 받지 못할 수도 있기 때문입니다. 충전 상태일 때는 배터리 잔량이 모자라더라도 재생을 진행해도 됩니다.

현재 배터리 상태를 확인하고 배터리 관련 이벤트를 구하는 방법을 알아보겠습니다.

1) 배터리 상태 정보 구하기

새로운 소스 프로젝트를 생성하고 Project name을 BatteryInfo 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 화면 위쪽에 라이브러리를 추가합니다.

```
#include "batteryinfo.h"  
#include <device/battery.h>  
#include <device/callback.h>
```

device/battery.h는 배터리 정보 라이브러리 헤더파일 입니다.

device/callback.h는 장비 관련 이벤트 콜백 라이브러리 헤더파일 입니다.

Button을 누르면 배터리 잔량과 충전 여부를 화면에 표시하는 기능을 구현해 보겠습니다. create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}

```

그런 다음 `create_base_gui()` 함수 끝부분에 Box와 Button 생성 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
/* Label*/
ad->label = elm_label_add(ad->win);
elm_object_text_set(ad->label, "<align=center>Hello Tizen</>");
/* expand horizontally but not vertically, and fill horiz,
 * align center vertically */
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.0);

/* Button-1 */
Evas_Object *btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Default style");
evas_object_smart_callback_add(btn, "clicked", show_battery_state, ad);
/* epand both horiz and vert, fill horiz and vert */
my_box_pack(box, btn, 1.0, 1.0, -1.0, 0.0);
}

}

```

```
/* Show window after base gui is set up */  
evas_object_show(ad->win);
```

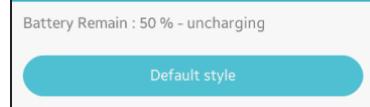
create_base_gui() 함수 위에 Button 콜백 함수를 추가합니다.

```
static void  
show_battery_state(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    int result=0, percent=0;  
    bool charging = false;  
    device_battery_get_percent(&percent);  
    device_battery_is_charging(&charging);  
    char buf[100];  
    sprintf(buf, "Battery Remain : %d %% - %s", percent, charging ? "charging" :  
"uncharging");  
  
    elm_object_text_set(ad->label, buf);  
}
```

device_battery_get_percent(int *) 는 배터리 잔량을 퍼센트 단위로
환산으로 반환하는 API 입니다.

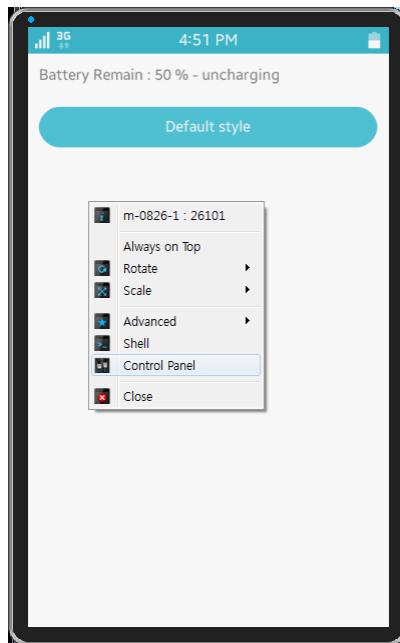
device_battery_is_charging(bool *) 는 충전 상태를 반환하는 API 입니다.
충전중인 경우에는 true를 반환하고, 충전 상태가 아니라면 false를
반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 배터리 잔량과 충전
상태가 Label 위젯에 표시됩니다. 에뮬레이터에서 실행했다면 잔량 50%,
uncharging 이라고 표시될 겁니다.



2) 에뮬레이터에서 배터리 상태 변경

에뮬레이터에서 배터리 상태를 변경하려면 Control Panel을 사용하면 됩니다. 에뮬레이터를 마우스 오른쪽 버튼으로 클릭하면 단축메뉴 [Control Panel]을 선택합니다.

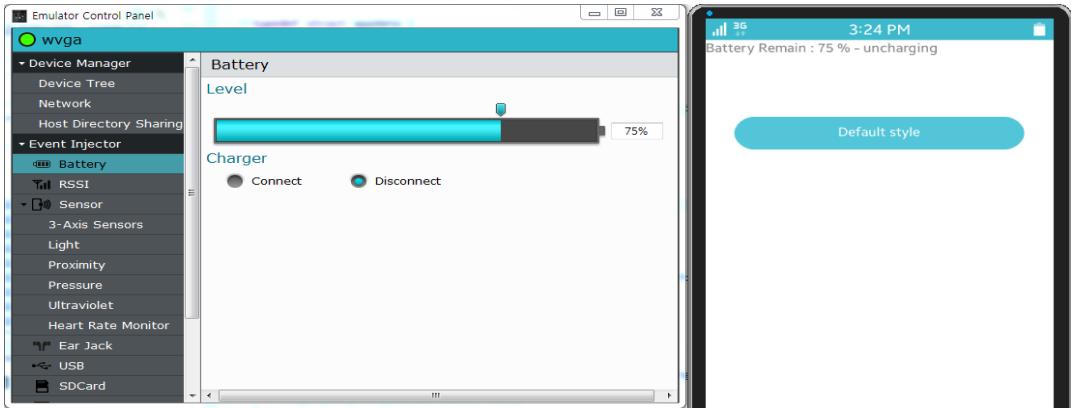


새로운 팝업창이 나타나면 왼쪽 트리 목록에서 [Event Injector > Battery]를 선택합니다.

오른쪽에 배터리 모양의 슬라이더가 나타나면 바를 드래그해서 수치를 변경한 다음, Charger 하위에 있는 Connect 라디오 버튼을 선택합니다.

그런 다음 에뮬레이터에서 'Default style' 버튼을 누르면 새로운 정보가

표시됩니다.



3) 배터리 상태 변경 이벤트 구하기

충전 케이블이 연결되면 자동으로 이벤트를 구하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
    evas_object_show(ad->win);

    device_add_callback(DEVICE_CALLBACK_BATTERY_CHARGING,
battery_charging_cb, ad);
}
```

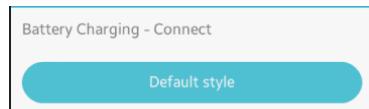
`device_add_callback(device_callback_e, device_changed_cb, void *)` 는 다바이스 이벤트의 콜백 함수를 지정하는 API입니다. 1번째 파라미터에 `DEVICE_CALLBACK_BATTERY_CHARGING`를 전달하면 충전기 연결 이벤트를 구할 수 있습니다. `DEVICE_CALLBACK_BATTERY_LEVEL`를 전달하면 배터리 잔량 변경 이벤트를 구할 수 있습니다.

`create_base_gui()` 함수 위에 콜백 함수를 만들 차례입니다.

```
static void battery_charging_cb(device_callback_e type, void *value, void *user_data)
{
    appdata_s *ad = user_data;
    char buf[100];
    sprintf(buf, "Battery Charging - %s", (int)value ? "Connect" : "Disconnect");
    elm_object_text_set(ad->label, buf);
}
```

충전기 이벤트 함수의 2번째 파라미터에 1이 전달되면 연결 이벤트이고, 0이 전달되면 연결 해제 이벤트입니다.

예제를 다시 실행하고 Control Panel에서 Charger 하위에 라디오 버튼을 번갈아서 눌러 봅니다. Label 위젯에 새로운 메시지가 표시됩니다.



4) 배터리 위험 수위 이벤트 구하기

배터리 잔량이 15% 이하로 떨어진 경우에는 절전모드로 전환해야 합니다. 소스파일 아래쪽으로 이동하면 ui_app_low_battery() 이라는 함수가 있는데 이것이 배터리 위험 수위를 알리는 이벤트 함수입니다. ui_app_low_battery() 함수에 새로운 코드를 추가합니다.

만약 이 함수가 없거나 함수명을 변경하고 싶으면 main() 함수에서 변경할 수 있습니다.

```
static void
```

```

ui_app_low_battery(app_event_info_h event_info, void *user_data)
{
    show_battery_state(user_data, NULL, NULL);
}

int
main(int argc, char *argv[])
{
    appdata_s ad = {0,};
    int ret = 0;

    ui_app.lifecycle_callback_s event_callback = {0,};
    app_event_handler_h handlers[5] = {NULL, };

    event_callback.create = app_create;
    event_callback.terminate = app_terminate;
    event_callback.pause = app_pause;
    event_callback.resume = app_resume;
    event_callback.app_control = app_control;

    ui_app_add_event_handler(&handlers[APP_EVENT_LOW_BATTERY],
APP_EVENT_LOW_BATTERY, ui_app_low_battery, &ad);
    ui_app_add_event_handler(&handlers[APP_EVENT_LANGUAGE_CHANGED],
APP_EVENT_LANGUAGE_CHANGED, ui_app_lang_changed, &ad);

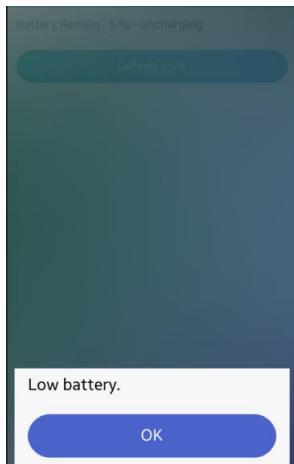
    ret = ui_app_main(argc, argv, &event_callback, &ad);
    if (ret != APP_ERROR_NONE) {
        dlog_print(DLOG_ERROR, LOG_TAG, "app_main() is failed. err = %d", ret);
    }

    return ret;
}

```

배터리가 위험 수위에 도달하면 자동으로 현재 배터리 상태를 화면에 표시합니다.

예제를 다시 실행시켜서 Control Panel에서 Disconnect에 체크하고 배터리 잔량을 15% 이하로 변경해 보겠습니다. 경고 팝업창이 나타나고 Label 위젯에 메시지가 변경됩니다.



5) 관련 API

device/battery.h : 배터리 정보 라이브러리 헤더파일.

device/callback.h ; 장비 관련 이벤트 콜백 라이브러리 헤더파일.

int device_battery_get_percent(int *percent) : 배터리 잔량을 퍼센트 단위로 환산으로 반환하는 API.

int device_battery_get_percent(int *percent) : 충전 상태를 반환하는 API. 충전중인 경우에는 true를 반환하고, 충전 상태가 아니라면 false를 반환합니다.

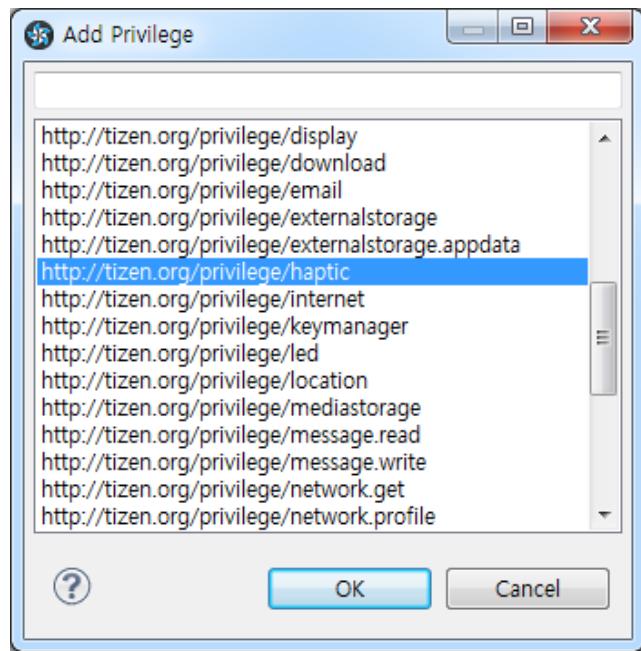
int device_add_callback(device_callback_e type, device_changed_cb callback, void *user_data) : 다바이스 이벤트의 콜백 함수를 지정하는 API. 1번째 파라미터에 DEVICE_CALLBACK_BATTERY_CHARGING를 전달하면 충전기 연결 이벤트를 구할 수 있습니다.
DEVICE_CALLBACK_BATTERY_LEVEL를 전달하면 배터리 잔량 변경 이벤트를 구할 수 있습니다.

45. 진동 발생

무음모드 일때 알림 메시지를 받으면 사용자에게 진동으로 알려야 합니다. 게임을 할 때 충격을 받거나 새로운 메시지를 수신 받았을 때 진동 효과를 주면 사용자가 현실감을 느끼게 됩니다. 만약 3D 그래픽과 결합하게 되면 4D가 구현되는 것입니다. 이번 예제에서는 진동을 발생하는 방법을 알아보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 VibrateEx 으로 지정합니다. 진동 기능을 사용하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/haptic>을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



저장하고 아래쪽 텁버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.vibrateex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application  appid="org.example.vibrateex"  exec="vibrateex"  multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>vibrateex</label>
        <icon>vibrateex.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/haptic</privilege>
    </privileges>
</manifest>
```

2) Haptic 장비 개수 구하기

Haptic 은 촉각 인터페이스입니다. 폰에 몇가지 Haptic 장비가 장착되어 있는지 개수를 구해보겠습니다. src 폴더에 소스파일(~.c)을 열고 소스파일 위쪽에 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "vibrateex.h"  
#include <device/haptic.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    haptic_device_h handle;  
    haptic_effect_h effect_handle;  
  
    Ecore_Timer *timer1;  
    int timer_count;  
} appdata_s;
```

device/haptic.h 는 Haptic 장비를 사용하기 위한 라이브러리입니다.

haptic_device_h 는 Haptic 장비를 제어할 수 있는 핸들입니다.

haptic_effect_h 는 한가지 Haptic 효과를 제어할 수 있는 핸들입니다.

create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */  
evas_object_show(ad->win);
```

```
    haptic_count(ad);  
}  
└─────────────────┘
```

haptic_count() 는 Haptic 장비 개수를 화면에 표시하는 함수입니다.
create_base_gui() 함수 위에 이 함수를 생성합니다.

```
└─────────────────┘  
static void  
haptic_count(appdata_s *ad)  
{  
    int error, num;  
    error = device_haptic_get_count(&num);  
  
    char buf[100];  
    sprintf(buf, "Haptic count : %d", num);  
    elm_object_text_set(ad->label, buf);  
}  
└─────────────────┘
```

device_haptic_get_count(int *) 는 Haptic 장비 개수를 반환하는 API입니다. 만약 1보다 작다면 진동 기능이 지원되지 않습니다.

예제를 빌드하고 스마트폰에 설치해 봅시다. 에뮬레이터에서는 진동 기능을 테스트 해볼수 없습니다. 예제가 실행되면 Haptic 장비 개수가 Label 위젯에 표시됩니다.



2) 진동 기능 발생

Haptic 객체를 생성해서 진동을 발생시켜 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수 2개를 생성합니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}
```

```

static Evas_Object *
my_button_add(Evas_Object *parent, const char *text, Evas_Smart_Cb cb, void *cb_data)
{
    Evas_Object *btn;

    btn = elm_button_add(parent);
    elm_object_text_set(btn, text);
    evas_object_smart_callback_add(btn, "clicked", cb, cb_data);

    return btn;
}

```

my_box_pack() 은 Box 컨테이너에 위젯을 추가하는 함수입니다.

my_button_add() 는 Button 위젯을 생성하는 함수입니다.

create_base_gui() 함수에 새로운 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

Evas_Object *btn, *box;

/* Container: standard table */
box = elm_box_add(ad->win);
elm_box_homogeneous_set(box, EINA_TRUE);

```

```

elm_box_horizontal_set(box, EINA_FALSE);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* Label*/
ad->label = elm_label_add(box);
my_box_pack(box, ad->label, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND, 0.5,
0.5);

/* Buttons */
btn = my_button_add(box, "Vibrate", btn_vibrate_cb, ad);
my_box_pack(box, btn, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND,
EVAS_HINT_FILL, EVAS_HINT_FILL);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

/* Haptic */
haptic_count(ad);
device_haptic_open(0, &ad->handle);
}

```

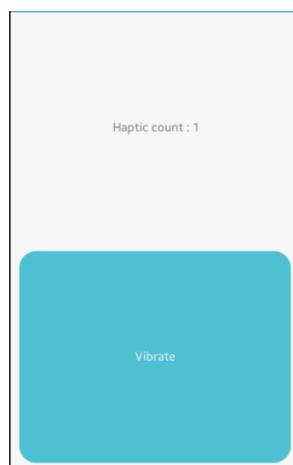
device_haptic_open(int, haptic_device_h *) 는 haptic 객체를 생성하는 API입니다. 1번째 파라미터에 Haptic 장비 번호를 전달하면 2번째 파라미터에선 Haptic 객체를 반환해 줍니다.

버튼을 누르면 5초 동안 진동이 발생하는 기능을 구현해 보겠습니다. create_base_gui() 함수 위에 Button 콜백 함수를 생성합니다.

```
static void  
btn_vibrate_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    int error = device_haptic_vibrate(ad->handle, 5000, 100, &ad->effect_handle);  
}
```

device_haptic_vibrate(haptic_device_h, int, int, haptic_effect_h *) 는 진동을 발생하는 API입니다. 1번째 파라미터는 Taptic 객체, 2번째는 지속 시간(단위는 밀리세컨), 3번째는 강도(0~100), 4번째 파라미터에는 Haptic 효과 제어 핸들을 반환해 줍니다. 진동을 강제 중지할 때 사용됩니다.

예제를 다시 실행하고 Button을 눌러봅시다. 5초 동안 진동이 발생됩니다.



3) 진동 기능 중지

2번째 Button을 추가해서 진동을 강제 중지하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 Button 생성 코드를 추가합니다.

```
/* Buttons */
btn = my_button_add(box, "Vibrate", btn_vibrate_cb, ad);
my_box_pack(box,     btn,     EVAS_HINT_EXPAND,     EVAS_HINT_EXPAND,
EVAS_HINT_FILL, EVAS_HINT_FILL);

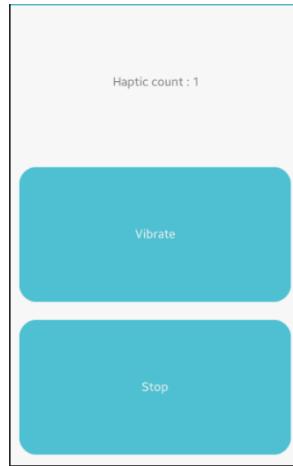
btn = my_button_add(box, "Stop", btn_stop_cb, ad);
my_box_pack(box,     btn,     EVAS_HINT_EXPAND,     EVAS_HINT_EXPAND,
EVAS_HINT_FILL, EVAS_HINT_FILL);
}
```

그런 다음 `create_base_gui()` 함수 위에 Button 콜백 함수를 생성합니다.

```
static void
btn_stop_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    int error = device_haptic_stop(ad->handle, &ad->effect_handle);
}
```

`device_haptic_stop(haptic_device_h, haptic_effect_h)` 는 Haptic 기능을 중지하는 API입니다. 1번째 파라미터에는 Haptic 객체, 2번째에는 효과 제어 핸들을 전달합니다.

예제를 다시 실행하고 1번째 Button을 눌러봅시다. 진동이 시작되면 2번째 Button을 누릅니다. 진동이 중지됩니다.



4) Dynamic Vibrate

타이머를 이용해서 진동 On/Off가 반복되는 기능을 구현해 보겠습니다.
create_base_gui() 함수에 3번째 Button 생성 코드를 추가합니다.

```
btn = my_button_add(box, "Stop", btn_stop_cb, ad);
my_box_pack(box,      btn,      EVAS_HINT_EXPAND,      EVAS_HINT_EXPAND,
EVAS_HINT_FILL, EVAS_HINT_FILL);

btn = my_button_add(box, "Dynamic Vibrate", btn_dynamic_cb, ad);
my_box_pack(box,      btn,      EVAS_HINT_EXPAND,      EVAS_HINT_EXPAND,
EVAS_HINT_FILL, EVAS_HINT_FILL);
}
```

마지막으로 create_base_gui() 함수 위에 새로운 함수 3개를 추가합니다.

```
static void
dynamic_vibrate(appdata_s *ad)
```

```

{
    if( ad->effect_handle != NULL )
        device_haptic_stop(ad->handle, &ad->effect_handle);

    if( (ad->timer_count % 2) == 0 )
        device_haptic_vibrate(ad->handle, 500, 100, &ad->effect_handle);
}

static Eina_Bool
timer1_cb(void *data EINA_UNUSED)
{
    appdata_s *ad = data;
    ad->timer_count++;
    dynamic_vibrate(ad);

    if(ad->timer_count > 5)
    {
        ecore_timer_del(ad->timer1);
        ad->timer1 = NULL;
    }
    return ECORE_CALLBACK_RENEW;
}

static void
btn_dynamic_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    ad->timer_count = 0;
    if (ad->timer1)
        ecore_timer_del(ad->timer1);
    ad->timer1 = ecore_timer_add(0.5, timer1_cb, ad);
    ad->effect_handle = NULL;

    dynamic_vibrate(ad);
}

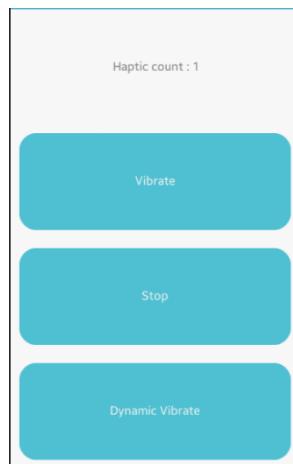
```

`dynamic_vibrate()` 는 0.5초 마다 호출되어서 진동 On/Off를 번갈아서 수행하는 함수입니다.

`timer1_cb()` 는 0.5초 마다 호출되는 타이머 이벤트 함수입니다. 3번째 Button을 누르면 이 함수가 총 6번 호출됩니다.

`btn_dynamic_cb()`는 전역변수를 초기화하고 타이머를 시작하는 함수입니다.

예제를 다시 실행하고 3번째 Button을 눌러봅시다. 0.5초 마다 진동 On/Off가 반복됩니다.



5) 관련 API

`int device_haptic_get_count(int *device_number)` : Haptic 장비 개수를 반환하는 API. 만약 1보다 작다면 진동 기능이 지원되지 않습니다.

`int device_haptic_open(int device_index, haptic_device_h *device_handle)` : haptic 객체를 생성하는 API. 1번째 파라미터에 Haptic 장비 번호를 전달하면 2번째 파라미터에선 Haptic 객체를 반환해

줍니다.

int device_haptic_vibrate(haptic_device_h device_handle, int duration, int feedback, haptic_effect_h *effect_handle) : 진동을 발생하는 API. 1번째 파라미터는 Taptic 객체, 2번째는 지속 시간(단위는 밀리세컨), 3번째는 강도(0~100), 4번째 파라미터에는 Haptic 효과 제어 핸들을 반환해 줍니다. 진동을 강제 중지할 때 사용됩니다.

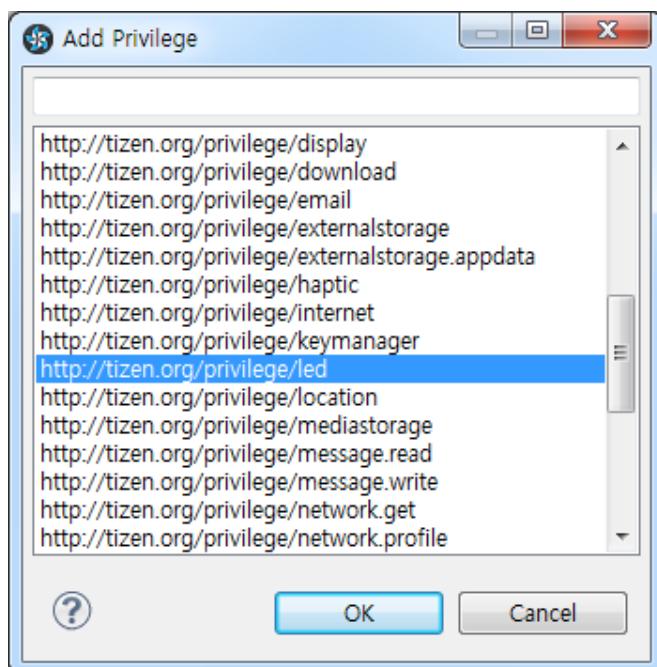
int device_haptic_stop(haptic_device_h device_handle, haptic_effect_h effect_handle) : Haptic 기능을 중지하는 API. 1번째 파라미터에는 Haptic 객체, 2번째에는 효과 제어 핸들을 전달합니다.

46. LED 플래쉬 백라이트

이번 예제에서는 카메라 플래쉬로 사용되는 LED 백라이트를 On/Off하는 방법을 알아보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 LedFreshEx 으로 지정합니다. 백라이트를 제어하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/led>을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



저장하고 아래쪽 탭버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.ledfreshex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.ledfreshex" exec="ledfreshex" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>ledfreshex</label>
        <icon>ledfreshex.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/led</privilege>
    </privileges>
</manifest>
```

2) LED 최대 밝기 구하기

LED를 On 할 때 밝기를 지정해야 합니다. 최대 밝기를 구해보겠습니다.
src 폴더에 소스파일(~.c)을 열고 소스파일 위쪽에 라이브러리
헤더파일과 변수를 추가합니다.

```
#include "ledfresh.h"
#include <device/led.h>

typedef struct appdata {
```

```
Evas_Object *win;
Evas_Object *conform;
Evas_Object *label;
int max;
} appdata_s;

```



device/led.h 는 LED 제어 라이브러리입니다.

max 는 최대 밝기를 저장하는 변수입니다.

create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

get_max_brightness(ad);
}

```



get_max_brightness 는 LED 백라이트의 최대 밝기를 구하는 함수입니다.
create_base_gui() 함수 위에 이 함수를 생성하겠습니다.

```
static void
get_max_brightness(appdata_s *ad)
{
    int error = device_flash_get_max_brightness(&ad->max);
    int val = 0;
    error = device_flash_get_brightness(&val);

    char buf[PATH_MAX];
    sprintf(buf, "Max brightness : %d / %d", ad->max, val);
}

```

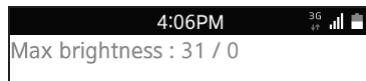


```
    elm_object_text_set(ad->label, buf);
}
```

device_flash_get_max_brightness(int *) 는 LED 백라이트의 최대 밝기를 구하는 API 입니다.

device_flash_get_brightness(int *) 는 LED 백라이트의 현재 밝기를 구하는 API 입니다.

예제를 실행시켜 봅시다. 최대 밝기와 현재 밝기 수치가 Label 위젯에 표시됩니다.



3) LED On/Off

2개의 Button을 추가해서 LED 백라이트 On/Off 기능을 구현해 보겠습니다. create_base_gui() 함수 위에 새로운 함수 2개를 생성합니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
```

```

/* set the input weight/aling on the frame insted of the child */
evas_object_size_hint_weight_set(frame, h_weight, v_weight);
evas_object_size_hint_align_set(frame, h_align, v_align);
{
    /* tell the child that is packed into the frame to be able to expand */
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    /* fill the expanded area (above) as opposaed to center in it */
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    /* actually put the child in the frame and show it */
    evas_object_show(child);
    elm_object_content_set(frame, child);
}
/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}

static Evas_Object *
my_button_add(Evas_Object *parent, const char *text, Evas_Smart_Cb cb, void *cb_data)
{
    Evas_Object *btn;

    btn = elm_button_add(parent);
    elm_object_text_set(btn, text);
    evas_object_smart_callback_add(btn, "clicked", cb, cb_data);

    return btn;
}

```

create_base_gui() 함수에 Button 위젯 생성 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

Evas_Object *btn, *box;

/* Container: standard table */
box = elm_box_add(ad->win);
elm_box_homogeneous_set(box, EINA_TRUE);
elm_box_horizontal_set(box, EINA_FALSE);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_size_hint_align_set(box, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    my_box_pack(box, ad->label, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND, 0.5,
0.5);

    /* Button-1 */
    btn = my_button_add(box, "LED On", btn_led_on_cb, ad);
    my_box_pack(box, btn, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND,
EVAS_HINT_FILL, EVAS_HINT_FILL);

    /* Button-2 */
    btn = my_button_add(box, "LED Off", btn_led_off_cb, ad);
    my_box_pack(box, btn, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
}

```

```
EVAS_HINT_FILL, EVAS_HINT_FILL);  
}  
  
/* Show window after base gui is set up */  
evas_object_show(ad->win);
```

그런 다음 create_base_gui() 함수 위에 Button 콜백 함수를 생성합니다.

```
static void  
btn_led_on_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    device_flash_set_brightness(ad->max);  
    device_led_play_custom(1000, 500, 0xFFFFF00, LED_CUSTOM_DEFAULT);  
}
```

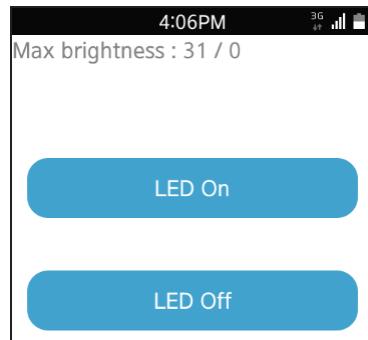
```
static void  
btn_led_off_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    device_led_stop_custom();  
}
```

device_flash_set_brightness(int) 는 LED 밝기를 지정하는 API 입니다.

device_led_play_custom(on, off, color, int) 는 LDE On 시작 API 입니다.

device_led_stop_custom(void) 는 LED Off API 입니다.

예제를 다시 실행시켜 봅시다. 'LED On' 버튼을 누르면 LED가 켜지고, 'LED Off' 버튼을 누르면 꺼집니다.



4) LED Off 오류 해결 방안

기종에 따라서 device_led_stop_custom() 함수를 사용해도 LED가 깨지지 않는 경우가 있습니다. 그런 경우에는 밝기를 최저로 낮추면 됩니다. btn_led_off_cb() 함수를 다음과 같이 수정합니다.

```
static void
btn_led_off_cb(void *data, Evas_Object *obj, void *event_info)
{
    device_flash_set_brightness(0);
    device_led_stop_custom();
}
```

device_flash_set_brightness() 함수를 사용해서 밝기를 0으로 지정하면 강제로 Off 시킬수 있습니다.

5) 관련 API

device/led.h : LED 제어 라이브러리.

int device_flash_get_max_brightness(int *max_brightness) : LED 백라이트의 최대 밝기를 구하는 API.

int device_flash_get_brightness(int *brightness) : LED 백라이트의 현재 밝기를 구하는 API.

int device_flash_set_brightness(int brightness) : LED 밝기를 지정하는 API.

int device_led_play_custom(int on, int off, unsigned int color, unsigned int flags) : LED On 시작 API.

int device_led_stop_custom(void) : LED Off API.

47. 화면 회전 방향 이벤트

폰의 회전 방향을 Orientation이라고 합니다. 폰이 수직방향인 경우를 Portrait라고 하고, 수평방향인 경우를 Landscape라고 합니다. 현재 폰의 화면 방향을 확인하고, Orientation 이벤트를 구하는 방법을 알아보겠습니다.

1) Orientation 방향 구하기

새로운 소스 프로젝트를 생성하고 Project name을 OrientationEvent로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

// Show now orientation
show_orientation(ad, NULL, NULL);

}
```

show_orientation() 은 현재 화면 방향을 구해서 Label 위젯에 출력하는 함수입니다. 지금부터 만들어 보겠습니다. create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```
// Show now orientation
static void
show_orientation(appdata_s *ad, Evas_Object *obj, void *event_info)
```

```

{
    // Get orientation
    int result = elm_win_rotation_get(ad->win);

    switch( result )
    {
        case APP_DEVICE_ORIENTATION_0 :
            elm_object_text_set(ad->label, "Portrait-1");
            break;
        case APP_DEVICE_ORIENTATION_90 :
            elm_object_text_set(ad->label, "Landscape-1");
            break;
        case APP_DEVICE_ORIENTATION_180 :
            elm_object_text_set(ad->label, "Portrait-2");
            break;
        case APP_DEVICE_ORIENTATION_270 :
            elm_object_text_set(ad->label, "Landscape-2");
            break;
        default :
            elm_object_text_set(ad->label, "Other Event");
            break;
    }
}

```

`show_orientation()` 는 Orientation 방향을 구해서 Label 위젯에 출력하는 함수입니다.

`elm_win_rotation_get(const Evas_Object *)` 는 현재 Orientation 방향을 반환하는 API입니다. 반환되는 종류는 다음과 같습니다.

- APP_DEVICE_ORIENTATION_0 : Portrait First
- APP_DEVICE_ORIENTATION_90 : Landscape First
- APP_DEVICE_ORIENTATION_180 : Portrait Second

- APP_DEVICE_ORIENTATION_270 : Landscape Second

예제를 빌드하고 실행시켜 봅시다. Label 위젯에 Portrait-1 이라는 텍스트가 표시됩니다.



2) 에뮬레이터에서 화면 회전

Button을 누르면 현재 Orientation 방향을 출력하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 생성합니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double
            v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
```

```

/* fill the expanded area (above) as opposed to center in it */
evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
/* actually put the child in the frame and show it */
evas_object_show(child);
elm_object_content_set(frame, child);
}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}

```

그런 다음 `create_base_gui()` 함수에 Button 생성 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
Evas_Object * box, *btn;

/* A box to put things in vertically - default mode for box */
box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

```

```

{ /* child object - indent to how relationship */
    /* Label*/
    ad->label = elm_label_add(ad->win);
    elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
    //evas_object_size_hint_weight_set(ad->label,      EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    //elm_object_content_set(ad->conform, ad->label);
    my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.0);

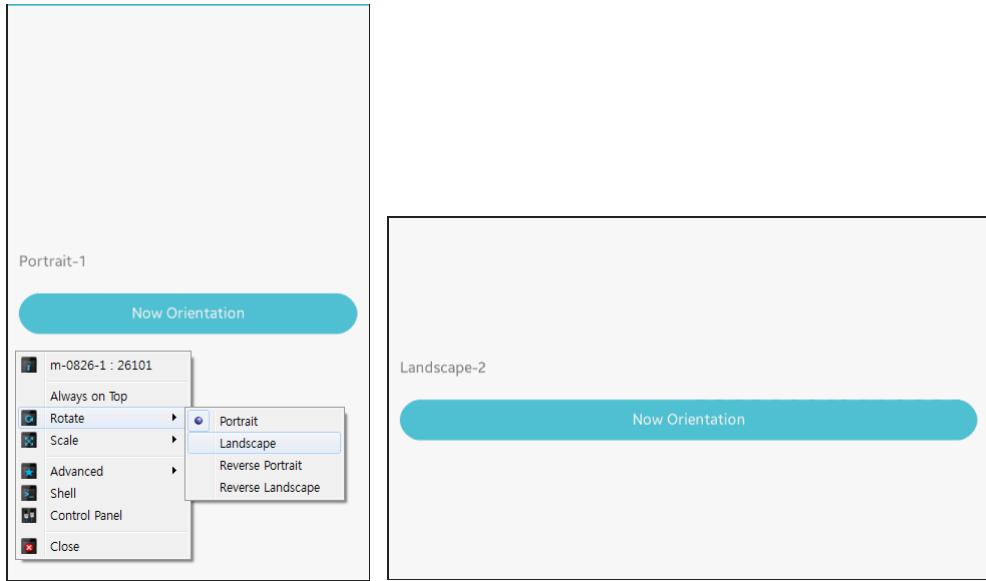
    /* Button-1 */
    btn = elm_button_add(ad->win);
    elm_object_text_set(btn, "Now Orientation");
    evas_object_smart_callback_add(btn, "clicked", show_orientation, (void
*)ad);
    my_box_pack(box, btn, 1.0, 0.0, -1.0, -1.0);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

에뮬레이터를 회전시켜 보겠습니다. 마우스 오른쪽 버튼으로
에뮬레이터를 클릭하고 단축메뉴에서 [Rotate > Landscape]를
선택합니다.



화면 방향이 회전하면 Button을 누릅니다. 이번에는 Landscape-2 라는 텍스트가 표시됩니다.

다시 화면 방향을 Portrait로 변경하고 Button을 누르면 Portrait-1 이 표시됩니다.

3) Orientation 방향 변경

이번에는 Button을 누르면 Orientation 방향이 변경되는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 2번째 Button 생성 코드를 추가합니다.

```
/* Button-1 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Now Orientation");
evas_object_smart_callback_add(btn, "clicked", show_orientation, (void *)ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, -1.0);
```

```

/* Button-2 */
btn = elm_button_add(ad->win);
elm_object_text_set(btn, "Orientation Change");
evas_object_smart_callback_add(btn,           "clicked",
btn_orientation_change_cb, (void *)ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, 0.0);

}

```

그런 다음, `create_base_gui()` 함수 위에 새로 추가된 Button의 콜백 함수를 생성합니다.

```

// 'Orientation Change' Button event function
static void
btn_orientation_change_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = (appdata_s*)data;
    // Get orientation
    int result = elm_win_rotation_get(ad->win);

    if( result == APP_DEVICE_ORIENTATION_0 || result == APP_DEVICE_ORIENTATION_180 )
        elm_win_rotation_with_resize_set(ad->win, APP_DEVICE_ORIENTATION_90);
    else
        elm_win_rotation_with_resize_set(ad->win, APP_DEVICE_ORIENTATION_0);
}

```

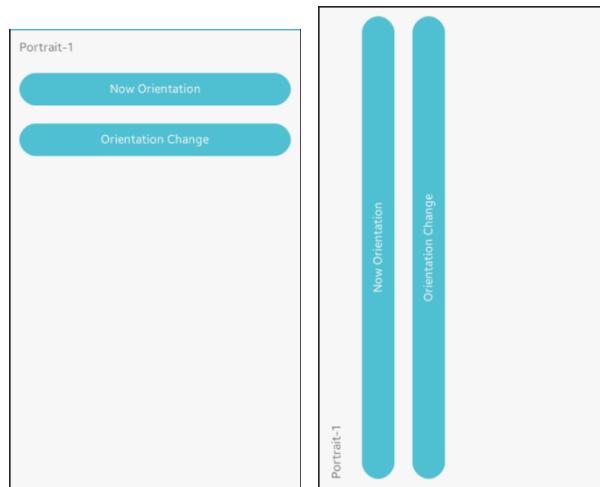
현재 Orientation 방향을 파악해서 Portrait 이면 Landscape로 변경하고, Landscape 면 Portrait로 변경하는 코드입니다.

`elm_win_rotation_with_resize_set(Evas_Object *, int)` 는 Orientation

방향을 변경하는 API입니다. 옵션 종류는 다음과 같습니다.

- APP_DEVICE_ORIENTATION_0 : Portrait First
- APP_DEVICE_ORIENTATION_90 : Landscape First
- APP_DEVICE_ORIENTATION_180 : Portrait Second
- APP_DEVICE_ORIENTATION_270 : Landscape Second

예제를 다시 실행하고 2번째 Button을 누릅니다. Button을 누를 때마다 Landscape 모드와 Portrait 모드가 반복됩니다.



4) Orientation 변경 이벤트 구하기

이번에는 Orientation 방향이 변경될 때의 이벤트를 구해 보겠습니다.

create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
// Show now orientation  
show_orientation(ad, NULL, NULL);  
  
// Set callback function of orientation change event  
evas_object_smart_callback_add(ad->win, "rotation_changed",
```

```
    win_rotation_changed_cb, ad);  
}  
_____
```

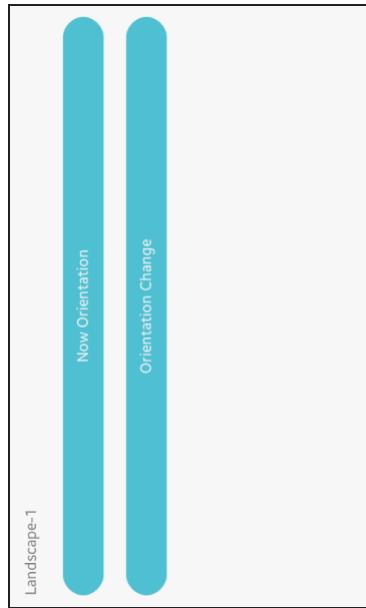
evas_object_smart_callback_add(Evas_Object *, char *, Evas_Smart_Cb, void *) 는 Layout 컨테이너 혹은 Button 위젯 같은 스마트 객체의 이벤트 콜백 함수를 지정하는 API입니다. 1번째 파라미터에 Win을 전달하고, 2번째 파라미터에 "rotation,changed"를 지정하면 Orientation 변경 이벤트를 구할 수 있습니다.

Orientation 변경 이벤트 함수를 만들 차례입니다. create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```
_____  
// Orientation changed event function  
static void  
win_rotation_changed_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = (appdata_s*)data;  
    show_orientation(ad, NULL, NULL);  
}
```

Orientation 방향이 변경되면 위 함수가 호출되고 현재 Orientation 정보를 구해서 화면에 표시합니다.

예제를 다시 실행하고 2번째 Button을 누릅니다. 화면 방향이 변경되면 자동으로 Label 위젯에 Landscape-1 이 표시됩니다.



5) Orientation 방향 고정

폰을 회전해도 Orientation 방향이 변경되지 않도록 고정시켜 보겠습니다. `create_base_gui()` 함수 시작부분에 허용 가능한 Orientation 종류가 정의되어 있습니다.

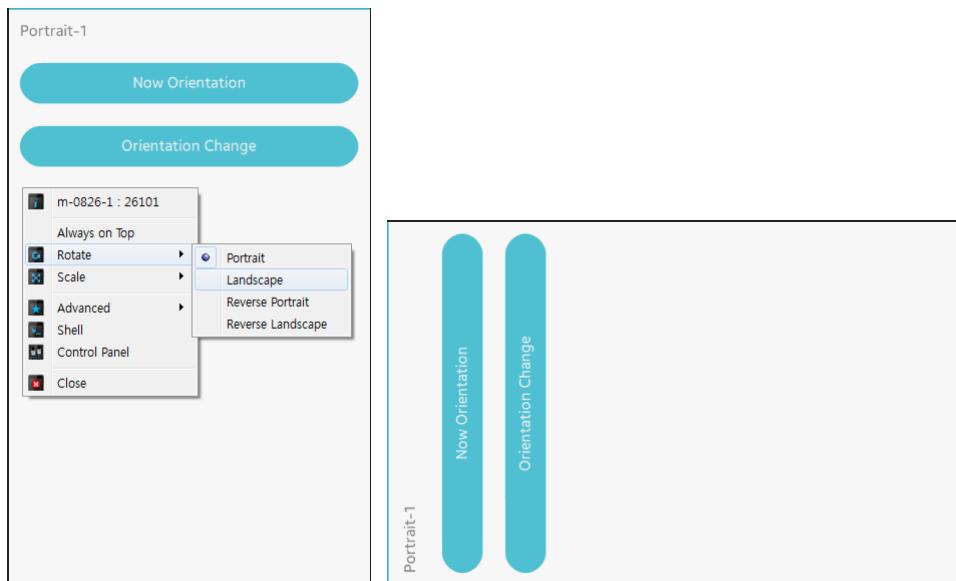
```
int rots[4] = { 0, 90, 180, 270 };
elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots), 4);
```

이 부분을 아래와 같이 변경합니다.

```
int rots[1] = { 0 };
elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots), 1);
```

`elm_win_wm_rotation_available_rotations_set(Evas_Object *, int *, unsigned int)` 는 허용 가능한 Orientation 방향 종류를 지정하는 API입니다. 2번째 파라미터에는 각도가 저장된 배열을 전달하고, 3번째 파라미터에는 배열에 저장된 데이터 개수를 전달합니다.

예제를 다시 실행하고 마우스 오른쪽 버튼을 클릭한 다음, 단축메뉴에서 [Rotate > Landscape]를 선택합니다. 에뮬레이터는 회전하지만 Orientation 모드는 변경되지 않습니다.



6) 관련 API

`int elm_win_rotation_get(const Evas_Object *obj)` : 현재 Orientation 방향을 반환하는 API. 반환되는 종류는 다음과 같습니다.

- APP_DEVICE_ORIENTATION_0 : Portrait First
- APP_DEVICE_ORIENTATION_90 : Landscape First
- APP_DEVICE_ORIENTATION_180 : Portrait Second
- APP_DEVICE_ORIENTATION_270 : Landscape Second

void elm_win_rotation_with_resize_set(Evas_Object *, int) : Orientation
방향을 변경하는 API. 옵션 종류는 다음과 같습니다.

- APP_DEVICE_ORIENTATION_0 : Portrait First
- APP_DEVICE_ORIENTATION_90 : Landscape First
- APP_DEVICE_ORIENTATION_180 : Portrait Second
- APP_DEVICE_ORIENTATION_270 : Landscape Second

void evas_object_smart_callback_add(Evas_Object *obj, const char
*event, Evas_Smart_Cb func, const void *data) : Layout 컨테이너 혹은
Button 위젯 같은 스마트 객체의 이벤트 콜백 함수를 지정하는 API.
1번째 파라미터에 Win을 전달하고, 2번째 파라미터에
"rotation,changed"를 지정하면 Orientation 변경 이벤트를 구할 수
있습니다.

void elm_win_wm_rotation_available_rotations_set(Evas_Object *obj,
const int *rotations, unsigned int count) : 허용 가능한 Orientation 방향
종류를 지정하는 API. 2번째 파라미터에는 각도가 저장된 배열을
전달하고, 3번째 파라미터에는 배열에 저장된 데이터 개수를 전달합니다.

48. 하드웨어 키 이벤트 & 디버그 모드

오디오 & 비디오 재생 앱 또는 게임이나 악기 앱을 제작할 경우, 볼륨 Up/Down 키로 볼륨 크기를 조절해야 합니다. 또한 하드웨어 Menu 버튼을 눌렀을 때 메뉴를 표시하거나 특정한 기능을 수행해야 합니다. 일반적으로 하드웨어 Back 버튼을 누르면 이전 화면으로 되돌아 가거나 앱을 종료하지만 경우에 따라서 경고 팝업창을 띄우거나 다른 기능을 수행해야 할 때가 있습니다. 이번 예제에서는 하드웨어 키 이벤트를 구하는 방법을 알아보겠습니다.

1) 하드웨어 키 이벤트 구하기

새로운 소스 프로젝트를 생성하고 Project name을 HardwareKeyEvent로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

/* Hardware key event callback */
evas_object_event_callback_add(ad->win,           EVAS_CALLBACK_KEY_DOWN,
on_keydown_cb, ad);
}
```

EVAS_CALLBACK_KEY_DOWN 는 하드웨어 키 다운 이벤트의 콜백 함수를 의미합니다.

이제 콜백 함수를 생성할 차례입니다. create_base_gui() 함수 위에

새로운 코드를 추가합니다.

```
static void
on_keydown_cb(void *data, Evas *evas, Evas_Object *o, void *event_info)
{
    appdata_s* ad = data;
    Evas_Event_Key_Down *ev = event_info;

    char *old_msg = elm_object_text_get(ad->label);
    char total_msg[PATH_MAX];
    char *key_value = strdup( ev->keyname );

    if( strcmp(ev->keyname, "XF86Menu") ==0 ) {
        key_value = "Menu";
    }
    else if( strcmp(ev->keyname, "XF86Home") ==0 ) {
        key_value = "Home";
    }
    else if( strcmp(ev->keyname, "XF86Back") ==0 ) {
        key_value = "Back";
    }
    else if( strcmp(ev->keyname, "XF86PowerOff") ==0 ) {
        key_value = "Power";
    }
    else if( strcmp(ev->keyname, "XF86AudioRaiseVolume") ==0 ) {
        key_value = "Volume Up";
    }
    else if( strcmp(ev->keyname, "XF86AudioLowerVolume") ==0 ) {
        key_value = "Volume Down";
    }
    else {
        key_value = ev->keyname;
    }
}
```

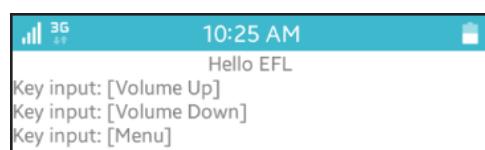
```
sprintf(total_msg, "%s<br/>Key input: [%s]", old_msg, key_value);
elm_object_text_set(ad->label, total_msg);
}
```

on_keydown_cb() 하드웨어 키 이벤트 함수입니다. 1번째 파라미터에는 사용자 데이터가 전달되고, 2번째에는 이벤트가 발생한 객체, 3번째에는 이벤트 정보가 전달됩니다.

이벤트 정보는 Evas_Event_Key_Down 형식으로 저장되어 있습니다. 속성 중에서 keyname에 문자열 형식의 키값이 저장되어 있습니다. 키값의 종류는 다음과 같습니다.

- XF86Menu : Menu 키
- XF86Home : Home 키
- XF86Back : Back 키
- XF86PowerOff : Power 키
- XF86AudioRaiseVolume : Volume Up 키
- XF86AudioLowerVolume : Volume Down 키

예제를 빌드하고 실행시켜 봅시다. 하드웨어 키 중에서 Volume Up 키와 Volume Down 키, 그리고 Menu 키를 눌러 봅시다. Label 위젯에 키 종류가 표시됩니다.

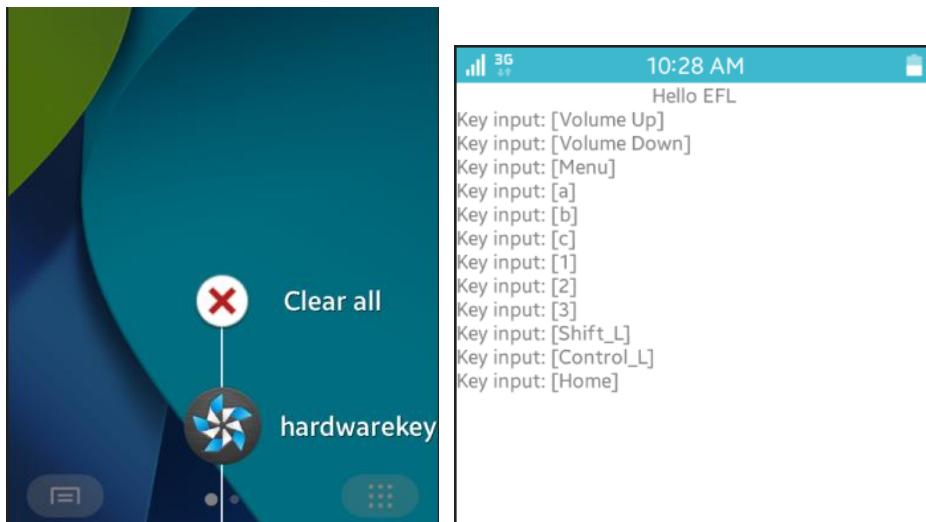


키보드에서 알파벳 키와 숫자 키, 그리고 Ctrl, Shift 키를 눌러봅시다.

해당 키값이 표시됩니다. 하드웨어 키패드가 장착된 폰의 경우에는 개별 키 입력을 구할 수 있습니다.

```
Hello EFL  
Key input: [Volume Up]  
Key input: [Volume Down]  
Key input: [Menu]  
Key input: [a]  
Key input: [b]  
Key input: [c]  
Key input: [1]  
Key input: [2]  
Key input: [3]  
Key input: [Shift_L]  
Key input: [Control_L]
```

이번에는 Home 키를 눌러봅시다. 앱이 사라집니다. 다시 Home 키를 오래 누르면 이제까지 실행했던 앱의 목록이 나타납니다. 그중에서 HardwareKeyEvent를 선택합니다. 예제가 다시 나타나고 Home 키가 인식된 것을 확인할 수 있습니다.



이번에는 Back 키를 눌러봅시다. 앱이 사라지면 Home 키를 오래 눌렀다가 앱 목록 중에서 HardwareKeyEvent를 선택합니다.

그 다음에는 Power 키를 눌러봅시다. 화면이 Off 되면 Home 키를 누르고, 화면을 드래그해서 잠금해제 합니다.



2) Log 메시지로 확인하기

Home 키, Back 키, Power 키의 경우에 곧바로 확인할 수 없다는 불편함이 있습니다. Log 메시지를 사용해서 확인해 보도록 하겠습니다. on_keydown_cb() 함수에 새로운 코드 1줄을 추가합니다.

```
static void
on_keydown_cb(void *data, Evas *evas, Evas_Object *o, void *event_info)
{
    appdata_s* ad = data;
    Evas_Event_Key_Down *ev = event_info;
    dlog_print(DLOG_INFO, "tag", ev->keyname);

    char *old_msg = elm_object_text_get(ad->label);
    char total_msg[PATH_MAX];
    char *key_value = strdup( ev->keyname );
```

키값을 Log 메시지로 출력하는 코드입니다.

이클립스 아래쪽 패널 중에서 Log를 선택합니다. 콤보박스에서 Tag를 선택하고 오른쪽 에디트박스에 tag라고 입력합니다.

예제를 다시 실행하고 하드웨어 키를 눌렀습니다. Home 키, Back 키, Power 키를 눌렀을 때 화면은 사라져도 메시지는 남습니다.

Time	Level	Pid	Tid	Tag	Message
07-24 10:31:50.852	Info	4913	4913	tag	XF86AudioRaiseVolume
07-24 10:31:55.272	Info	4913	4913	tag	XF86AudioLowerVolume
07-24 10:31:59.712	Info	4913	4913	tag	XF86Menu
07-24 10:32:02.032	Info	4913	4913	tag	XF86Home
07-24 10:32:19.831	Info	4913	4913	tag	XF86Back
07-24 10:32:26.911	Info	4913	4913	tag	XF86PowerOff

3) 디버그 모드

디버그 모드를 사용해서 특정 소스코드 위치에서 동작을 브레이크 지정하고, 변수의 값을 실시간으로 확인하는 방법을 알아보겠습니다.

on_keydown_cb() 함수의 다음 코드 왼쪽 에디트창 경계선을 마우스 클릭합니다. 그러면 하늘색 동그라미가 표시됩니다. 이것이 브레이크 포인트입니다.

```
char *key_value = strdup( ev->keyname );
```

```

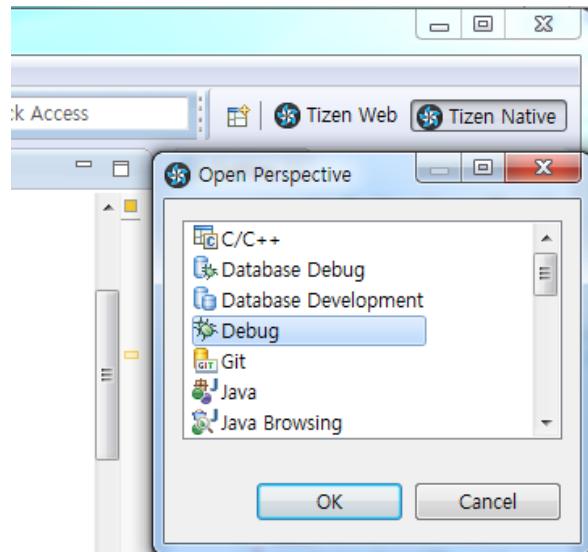
static void
on_keydown_cb(void *data, Evas *evas, Evas_Object *o, void *event_info)
{
    appdata_s* ad = data;
    Evas_Event_Key_Down *ev = event_info;
    dlog_print(DLOG_INFO, "tag", ev->keyname);

    char *old_msg = elm_object_text_get(ad->label);
    char total_msg[PATH_MAX];
    char *key_value = strdup( ev->keyname );

    if( strcmp(ev->keyname, "XF86Menu") ==0 ) {
        key_value = "Menu";
    }
}

```

디버그 모드로 변경해 보겠습니다. 이클립스 오른쪽 위 탭버튼 중에서 Debug를 선택합니다. Debug라는 탭버튼이 보이지 않을 때는 Open Perspective를 선택하고 팝업창이 나타나면 목록 중에서 Debug를 선택하면 됩니다.



이제 디버깅을 시작해 보겠습니다. 일반적으로 앱을 실행할 때는 Ctrl + F11을 사용하지만 디버깅을 할 때는 F11을 누릅니다. 또는 메인메뉴 [Run > Debug]를 선택해도 됩니다.

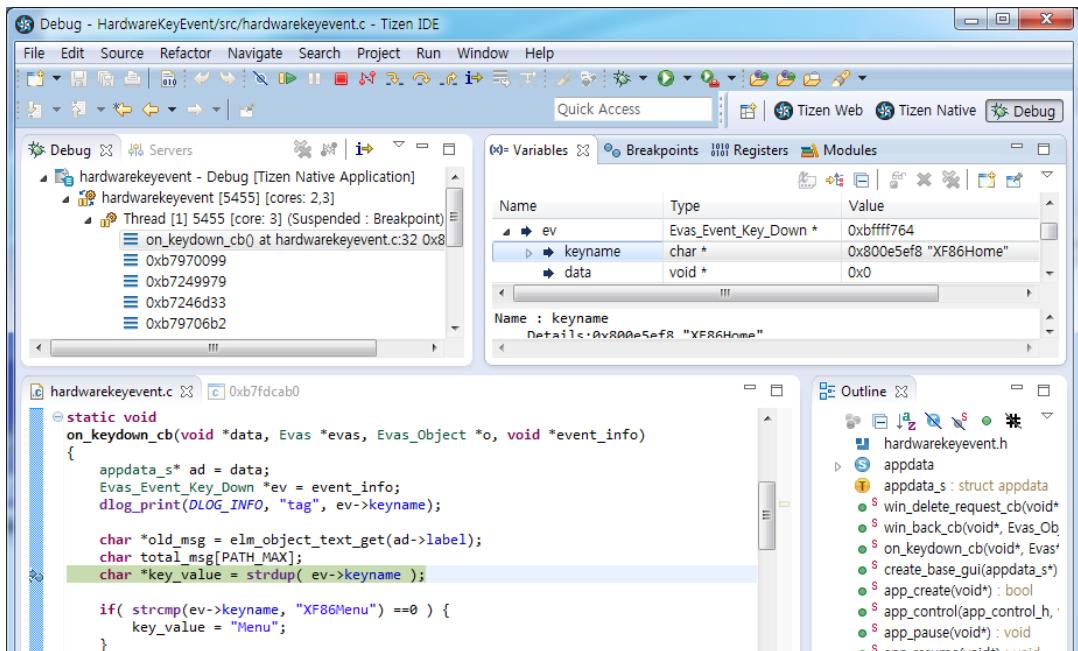
앱이 실행되다가 main() 함수에서 멈추거나 한다면 F8을 눌러서 다음 Step으로 넘어갑니다.

실행이 되었으면 Menu 키를 눌러봅시다. 브레이크 포인트에 화살표가 나타나고 동작이 멈춥니다.

이클립스 오른쪽 위 Variables 패널에는 변수 목록이 나타납니다. 그 중에서 ev를 선택해서 트리 목록을 엽니다.

하위 속성 중에서 keyname의 오른쪽에 키값이 보입니다

계속 진행하려면 F8을 누르면 다음 Step 으로 이동합니다.



디버그 모드에서 사용하는 단축키는 다음과 같습니다.

- F11 : 디버깅 시작
- F8 : Next step
- F5 : Step into
- F6 : Step over
- F7 : Step return
- Ctrl + F2 : 디버깅 중지

4) 관련 API

void evas_object_event_callback_add(Evas_Object *obj, Evas_Callback_Type type, Evas_Object_Event_Cb func, void *data) 함수의 2번째 파라미터에 EVAS_CALLBACK_KEY_DOWN를 전달하면 하드웨어 키다운 이벤트를 구할 수 있습니다.

하드웨어 키 이벤트 정보는 Evas_Event_Key_Down 형식으로 저장되어 있습니다. 속성 중에서 keyname에 문자열 형식의 키값이 저장되어 있습니다. 키값의 종류는 다음과 같습니다.

- XF86Menu : Menu 키
- XF86Home : Home 키
- XF86Back : Back 키
- XF86PowerOff : Power 키
- XF86AudioRaiseVolume : Volume Up 키
- XF86AudioLowerVolume : Volume Down 키

int dlog_print(log_priority prio, const char *tag, const char *fmt, ...) : 실시간으로 Log 메시지 출력 API.

디버그 모드에서 사용하는 단축키는 다음과 같습니다.

- F11 : 디버깅 시작
- F8 : Next step
- F5 : Step into
- F6 : Step over
- F7 : Step return
- Ctrl + F2 : 디버깅 중지

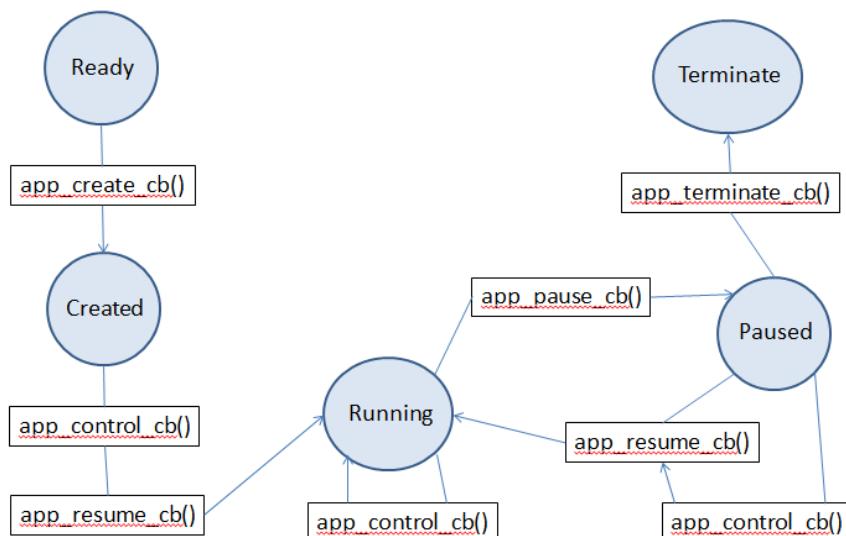
49. 라이프 사이클 & 디버그 모드

앱이 처음 실행될 때 UI를 구성하는 컨테이너와 위젯을 생성합니다. 변수에 메모리를 할당하는 작업은 이때 해주면 편리합니다. 앱이 종료될 때에는 메모리를 삭제해야 합니다. 배경음악을 재생해야 한다면 앱이 처음 실행될 때 음악을 Play하고, 종료될 때 Stop해야 합니다.

앱이 백그라운드로 숨을 때는 화면에 표시되는 애니메이션을 중지해야 CPU 부하를 줄일 수 있습니다. 백그라운드에서 포그라운드로 전환될 때 애니메이션을 재시작해야 합니다. 이번 예제에서는 수명주기에 대해서 알아보겠습니다.

1) 수명주기 관련 이벤트

타이젠 네이티브 앱의 수명주기는 다음 그림을 참조하면 됩니다.



앱이 실행될 때 일반적인 순서는 `app_create_cb() => app_control_cb()`

=> app_resume_cb() 함수를 거치게 됩니다.

앱이 종료될 때는 app_terminate_cb() 함수가 호출됩니다.

백그라운드 모드로 전환되는 경우 처럼 앱이 일시정지 될 때는 app_pause_cb() 함수가 호출됩니다.

일시정지가 해제될 때는 app_resume_cb() 함수가 호출됩니다.

다른 앱에서 앱 Launch 요청이 발생하면 app_control_cb() 함수가 호출됩니다.

2) 수명주기 관련 이벤트 함수

새로운 소스 프로젝트를 생성하고 Project name을 LifeCycle 로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 소스파일 위부분에 새로운 코드를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
} appdata_s;

static void
show_message(appdata_s* ad, const char* msg)
{
    dlog_print(DLOG_ERROR, "tag", msg);
    char *old_msg = elm_object_text_get(ad->label);
    char total_msg[PATH_MAX];
```

```

sprintf(total_msg, "%s<br/>%s", old_msg, msg);
elm_object_text_set(ad->label, total_msg);

}

static void
win_back_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    show_message(ad, "win_back_cb()");
    /* Let window go to hide state. */
    elm_win_iconified_set(ad->win, EINA_TRUE);
}

```

`show_message()` 는 Label 위젯에 새로운 메시지를 추가하는 함수입니다. 어디에서나 호출하기 위해서 가장 위쪽에 배치하였습니다.

`win_back_cb()` 는 사용자가 Back 버튼을 눌렀을 때 실행되는 이벤트 콜백 함수입니다. 1번째 파라미터에는 appdata 객체가 전달됩니다.

소스파일 아래쪽에는 여러 가지 이벤트 콜백 함수들이 정의되어 있습니다. 이중에서 수명주기와 관련된 함수에 코드를 추가합니다.

```

static bool
app_create(void *data)
{
    appdata_s *ad = data;
    create_base_gui(ad);
    show_message(ad, "app_create()");
    return true;
}

static void

```

```

app_control(app_control_h app_control, void *data)
{
    appdata_s *ad = data;
    show_message(ad, "app_control()");
}

static void
app_pause(void *data)
{
    appdata_s *ad = data;
    show_message(ad, "app_pause()");
}

static void
app_resume(void *data)
{
    appdata_s *ad = data;
    show_message(ad, "app_resume()");
}

static void
app_terminate(void *data)
{
    appdata_s *ad = data;
    show_message(ad, "app_terminate()");
}

```

app_create() 는 앱이 생성될 때는 실행되는 콜백 함수입니다. UI 객체를 생성하는 create_base_gui() 함수를 여기에서 호출합니다.

app_control() 는 다른 앱에서 앱 Launch 요청이 발생했을 때 실행되는 콜백 함수입니다.

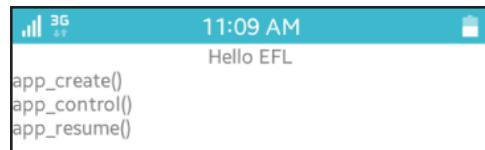
`app_pause()` 는 앱이 백그라운드 모드로 전환 될 때 실행되는 콜백 함수입니다.

`app_resume()` 는 앱이 백그라운드 모드에서 포그라운드 모드로 전환될 때 실행되는 콜백 함수입니다.

`app_terminate()` 는 앱이 종료될 때 실행되는 콜백 함수입니다.

이 5개의 콜백 함수는 `main()` 함수에서 지정합니다. 콜백 함수를 변경할 때는 `main()` 함수에서 해주면 됩니다.

예제를 빌드하고 실행시켜 봅시다. 앱이 실행되면 3가지 함수가 실행된 것을 확인할 수 있습니다. 앱이 실행된 순서는 `app_create() => app_control() => app_resume()` 입니다.

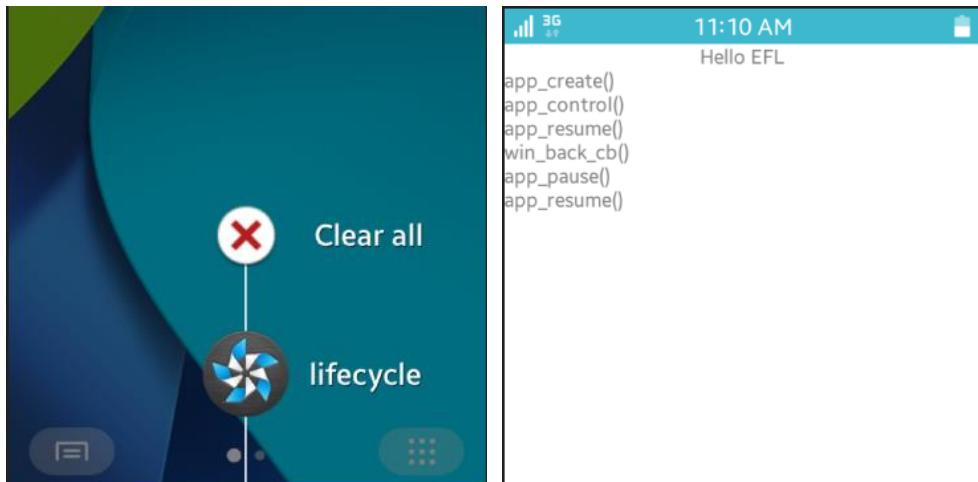


앱이 실행되었으면 Back 버튼을 눌러봅시다. 앱은 사라졌지만 종료된 것은 아닙니다. 백그라운드 모드에서 계속 실행되고 있습니다.

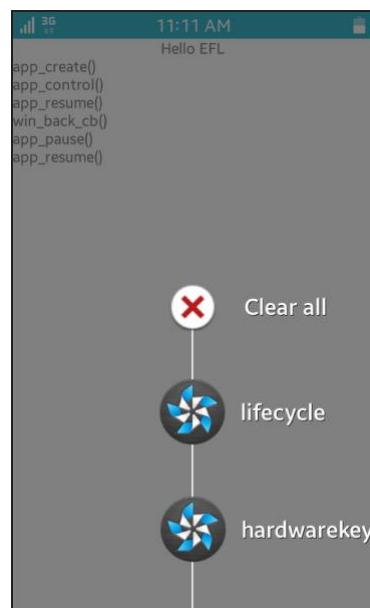
앱을 다시 불러오겠습니다. Home 버튼을 오래 누르고 있으면 이제까지 실행했던 앱의 목록이 나타납니다. 그 중에서 LifeCycle을 선택합니다.

예제가 다시 나타나고 3가지 함수가 추가로 실행된 것을 확인할 수 있습니다. 앱이 실행된 순서는 `win_back_cb() => app_pause() => app_resume()`입니다. Back 버튼을 눌렀을 때 `win_back_cb()` 함수가

실행되고, 앱이 백그라운드 모드로 전환될 때 app_resume() 함수가 실행됩니다. 포그라운드 모드로 전환될 때 app_resume() 함수가 실행됩니다.



이번에는 앱을 종료시켜 보겠습니다. Home 버튼을 오래 누르면 앱 목록이 나타납니다. Clear all을 눌러서 모든 앱을 종료합니다.



앱이 종료 되었습니다. 새로 추가된 메시지는 순식간에 지나가서 확인할 수 없습니다. 그러나 우리에게는 실시간 Log 메시지가 있습니다. 이클립스 아래쪽 Log 패널을 선택하고, 콤보박스에 Tag를 선택합니다. 그런 다음 오른쪽 에디트박스에 tag라고 입력합니다.

Label 위젯에 출력된 메시지가 보입니다. 가장 아래쪽에 app_terminate() 가 추가된 것을 확인할 수 있습니다. 앱이 종료될 때 app_terminate() 함수가 실행된 것입니다.

Time	Level	Pid	Tid	Tag	Message
06-29 11:10:35.365	Error	2631	2631	tag	app_create()
06-29 11:10:35.455	Error	2631	2631	tag	app_control()
06-29 11:10:35.455	Error	2631	2631	tag	app_resume()
06-29 11:20:25.478	Error	2631	2631	tag	win_back_cb()
06-29 11:20:25.988	Error	2631	2631	tag	app_pause()
06-29 11:22:42.276	Error	2631	2631	tag	app_resume()
06-29 11:26:17.883	Error	2631	2631	tag	app_terminate()

3) 디버그 모드

디버그 모드를 사용해서 특정 소스코드 위치에서 동작을 브레이크 지정하고, 변수의 값을 실시간으로 확인하는 방법을 알아보겠습니다.

win_back_cb() 함수로 이동해서 show_message() 호출 코드 왼쪽 에디트창 경계선을 마우스 클릭합니다. 그러면 하늘색 동그라미가 표시됩니다. 이것이 브레이크 포인트입니다.

```
static void  
win_back_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    show_message(ad, "win_back_cb()");  
    /* Let window go to hide state. */  
    elm_win_lower(ad->win);  
}
```

다음 5개의 함수에도 마찬가지로 show_message() 호출 코드에 브레이크 포인트를 지정합니다.

- app_create()
- app_control()
- app_pause()
- app_resume()
- app_terminate()

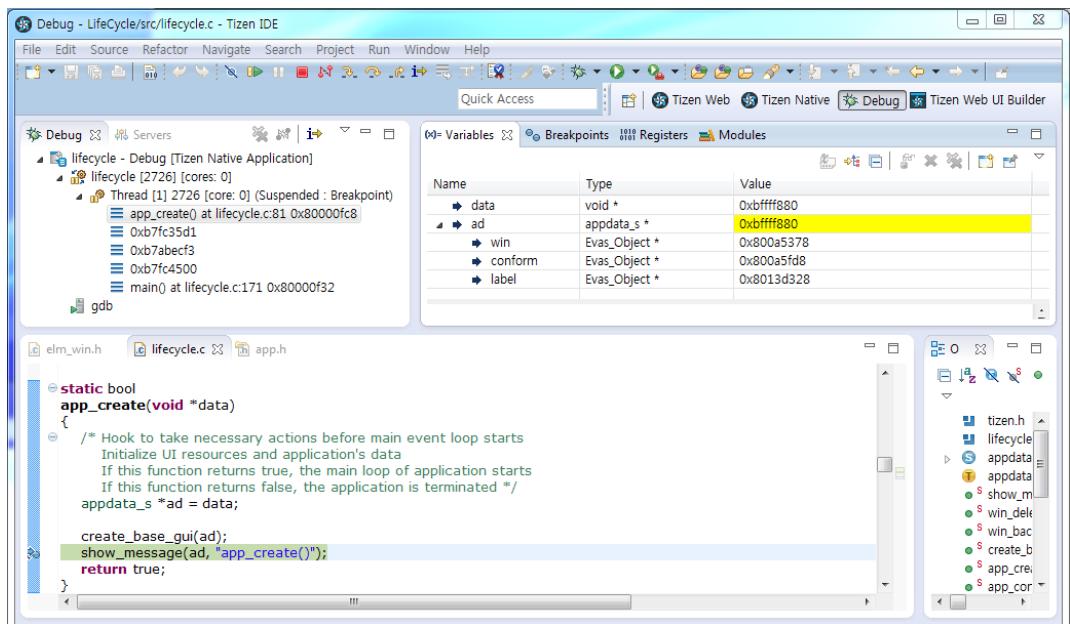
디버그 모드로 변경해 보겠습니다. 이클립스 오른쪽 위 탭버튼 중에서 Debug를 선택합니다.

이제 디버깅을 시작해 보겠습니다. 일반적으로 앱을 실행할 때는 Ctrl + F11을 사용하지만 디버깅을 할 때는 F11을 누릅니다.

앱이 실행되다가 main() 함수에 멈추거나 한다면 F8을 눌러서 다음 Step 으로 넘어갑니다.

앱이 실행되면 app_create() 함수에서 멈춥니다. F8을 누르면 브레이크 포인트를 지나서 app_control() 함수에서 다시 멈춥니다.

그 다음에는 app_resume() 함수에서 멈추고, 한번더 F8을 누르면 앱이 화면에 나타납니다.



Back 버튼을 누르면 win_back_cb() 함수에서 멈춥니다. F8을 누르면 app_pause()에서 멈춥니다.

다시 F8을 누르면 앱이 화면에서 사라집니다.

Home 키를 오래 눌렀다가 앱 목록에서 LifeCycle을 선택하면 app_resume() 함수에서 멈춥니다.

F8을 누르면 앱이 다시 화면에 나타납니다.

디버그 모드에서 사용하는 단축키는 다음과 같습니다.

- F11 : 디버깅 시작
- F8 : Next step
- F5 : Step into
- F6 : Step over
- F7 : Step return
- Ctrl + F2 : 디버깅 중지

4) 관련 API

void win_back_cb(void *data, Evas_Object *obj, void *event_info) : 사용자가 Back 버튼을 눌렀을 때 실행되는 이벤트 콜백 함수. 1번째 파라미터에는 appdata 객체가 전달됩니다.

bool app_create(void *data) : 앱이 생성될 때는 실행되는 콜백 함수. UI 객체를 생성하는 create_base_gui() 함수를 여기에서 호출합니다.

void app_control() : 다른 앱에서 앱 Launch 요청이 발생했을 때 실행되는 콜백 함수.

void app_pause() : 앱이 백그라운드 모드로 전환 될 때 실행되는 콜백 함수.

void app_resume() : 앱이 백그라운드 모드에서 포그라운드 모드로 전환될 때 실행되는 콜백 함수.

void app_terminate() : 앱이 종료될 때 실행되는 콜백 함수.

디버그 모드에서 사용하는 단축키는 다음과 같습니다.

- F11 : 디버깅 시작
- F8 : Next step
- F5 : Step into
- F6 : Step over
- F7 : Step return
- Ctrl + F2 : 디버깅 중지

50. Notify 사용방법

새로운 메시지가 수신되었을 때 사용자에게 메시지를 전달하려면 Notify를 사용하면 됩니다. 일정 시간이 지나면 자동으로 메시지가 사라지게 할 수 있고, Button 같은 위젯을 추가할 수도 있습니다. 사용자의 UI 이벤트를 차단할 수도 있으며, 아래쪽에 표시하는 것도 가능합니다.

1) Notify에 Timeout 지정

새로운 소스 프로젝트를 생성하고 Project name을 NotifyEx로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 `create_base_gui()` 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
}
```

```

/* tell the child that is packed into the frame to be able to expand */
evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

/* fill the expanded area (above) as opposed to center in it */
evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
/* actually put the child in the frame and show it */
evas_object_show(child);
elm_object_content_set(frame, child);

}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);

}

```

그런 다음 `create_base_gui()` 함수에 새로운 코드를 추가합니다. Box 컨테이너와 Button 그리고 Notify 객체를 생성하는 코드입니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

```

```

elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* Label */
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
    my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

    Evas_Object* notify = create_notify_top_timeout(box);

    /* Button-1 */
    Evas_Object *btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "Top / Time out");
    evas_object_smart_callback_add(btn, "clicked", btn_click_cb, notify);
    my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

create_notify_top_timeout() 는 Notify를 생성하고 Timeout을 지정하는 함수입니다. 지금부터 만들어 보겠습니다. create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```

static Evas_Object*
create_notify_top_timeout(Evas_Object *parent)
{
    Evas_Object *notify;
    Evas_Object *box;
    Evas_Object *label;
```

```

/* Create notify (top-aligned / hide automatically) */
notify = elm_notify_add(parent);
elm_notify_align_set(notify, 0.5, 0.0);
elm_notify_timeout_set(notify, 3.0);

/* Create box for stacking notify message */
box = elm_box_add(notify);
evas_object_show(box);

/* Create label for notify message */
label = elm_label_add(box);
evas_object_size_hint_min_set(label, ELM_SCALE_SIZE(480), 0);
elm_label_line_wrap_set(label, ELM_WRAP_WORD);
elm_object_text_set(label, "<font align=center>This notification will hide automatically
in 3 seconds later.</font>");
elm_box_pack_end(box, label);
evas_object_show(label);

elm_object_content_set(notify, box);
return notify;
}

static void
btn_click_cb(void *data, Evas_Object *obj, void *event_info)
{
    Evas_Object *notify = data;
    evas_object_show(notify);
}

```

`elm_notify_add(Evas_Object *)` 는 Notify 객체를 생성하는 API 입니다.

`elm_notify_align_set(Evas_Object *, double, double)`는 Notify의 위치를

비율 단위로 지정하는 API입니다. 2번째 파라미터는 수평 위치를 지정합니다. 0이면 왼쪽, 0.5면 가운데, 1이면 오른쪽에 배치됩니다. 3번째 파라미터는 수직 위치를 지정합니다. 0이면 위쪽, 0.5면 중앙, 1이면 아래쪽에 배치됩니다.

`elm_notify_timeout_set(Evas_Object *, double)` 는 Notify에 Timeout을 지정하는 API입니다. 2번째 파라미터에 시간 간격을 지정하며 단위는 초입니다.

그 다음 코드는 Notify에 Box 컨테이너를 생성하고, Box 컨테이너에 Label 위젯을 생성하는 코드입니다.

`btn_click_cb()` 는 사용자가 Button을 눌렀을 때 Notify를 화면에 표시하는 함수입니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 화면 위쪽에 Notify가 나타났다가 3초 후에 사라집니다.



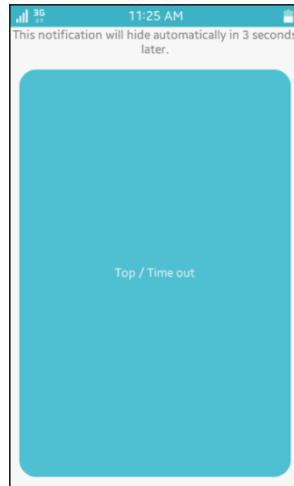
2) Notify 사이즈 조정

Notify에 표시되는 텍스트 문자열이 길어서 앞뒤가 잘리는 현상이 나타납니다. App base scale을 변경해서 이 현상을 해결하겠습니다. app_create() 함수에 새로운 코드 1줄을 추가합니다.

```
static bool  
app_create(void *data)  
{  
    appdata_s *ad = data;  
    elm_app_base_scale_set(1.8);  
    create_base_gui(ad);  
  
    return true;  
}
```

elm_app_base_scale_set(double) 는 App base scale을 변경하는 API입니다. 기본 Scale은 1.0입니다. Scale 수치가 클수록 Notify의 넓이가 줄어듭니다.

예제를 다시 실행하고 Button을 눌러봅시다. 이번에는 Notify의 모든 텍스트가 제대로 보입니다.



3) Notify에 Button 위젯 추가

Notify에 Button을 추가해서 Timeout 대신에 Button을 누르면 Notify가 사라지는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* Button-1 */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Top / Time out");
evas_object_smart_callback_add(btn, "clicked", btn_click_cb, notify);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

notify = create_notify_top_manual(box);

/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Top / Manual");
evas_object_smart_callback_add(btn, "clicked", btn_click_cb, notify);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
```

2번째 notify와 2번째 Button을 생성하는 코드입니다. create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

```
static void
btn_hide_notify_cb(void *data, Evas_Object *obj, void *event_info)
{
    Evas_Object *notify = data;
    evas_object_hide(notify);
}

static Evas_Object*
create_notify_top_manual(Evas_Object *parent)
{
    Evas_Object *notify;
    Evas_Object *box;
    Evas_Object *label;
    Evas_Object *btn;

    /* Create notify (top-aligned / hide manually) */
    notify = elm_notify_add(parent);
    elm_notify_align_set(notify, 0.5, 0.0);
    elm_notify_timeout_set(notify, 0.0);

    /* Create box for stacking notify message and button vertically */
    box = elm_box_add(notify);
    elm_box_horizontal_set(box, EINA_FALSE);
    evas_object_show(box);

    /* Create label for notify message */
    label = elm_label_add(box);
    evas_object_size_hint_min_set(label, ELM_SCALE_SIZE(480), 0);
    elm_label_line_wrap_set(label, ELM_WRAP_WORD);
    elm_object_text_set(label, "Click OK button to hide");
}
```

```

notification</center>");

elm_box_pack_end(box, label);
evas_object_show(label);

/* Create button to hide notify */
btn = elm_button_add(box);
elm_object_text_set(btn, "OK");
evas_object_size_hint_min_set(btn, ELM_SCALE_SIZE(80), ELM_SCALE_SIZE(58));
elm_box_pack_end(box, btn);
evas_object_show(btn);
evas_object_smart_callback_add(btn, "clicked", btn_hide_notify_cb, notify);

elm_object_content_set(notify, box);

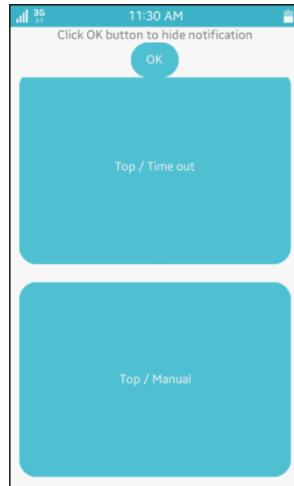
return notify;
}

```

`btn_hide_notify_cb()` 는 Notify를 감추는 함수입니다. Notify에 추가된 Button을 누르면 이 함수가 호출됩니다.

`create_notify_top_manual()` 는 2번째 Notify 객체를 생성하는 함수입니다. Notify에 Box 컨테이너를 생성하고, Box 컨테이너에 Label 위젯과 Button 위젯을 생성하는 코드입니다.

예제를 다시 실행하고 2번째 Button을 눌러봅시다. Notify가 나타나고 Label과 Button이 추가되어 있습니다. Button을 누르면 Notify가 사라집니다.



4) 이벤트 차단 Notify

Notify가 표시된 상태에서는 사용자의 이벤트를 차단하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Top / Manual");
evas_object_smart_callback_add(btn, "clicked", btn_click_cb, notify);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

notify = create_notify_top_block(box);

/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Top / Block");
evas_object_smart_callback_add(btn, "clicked", btn_click_cb, notify);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
```

3번째 notify와 3번째 Button을 생성하는 코드입니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```
static Evas_Object*
create_notify_top_block(Evas_Object *parent)
{
    Evas_Object *notify;
    Evas_Object *box;
    Evas_Object *label;

    /* Create notify (top-aligned / hide automatically / block outside events) */
    notify = elm_notify_add(parent);
    elm_notify_align_set(notify, 0.5, 0.0);
    elm_notify_timeout_set(notify, 3.0);
    elm_notify_allow_events_set(notify, EINA_FALSE);

    /* Create box for stacking notify message */
    box = elm_box_add(notify);
    evas_object_show(box);

    /* Create label for notify message */
    label = elm_label_add(box);
    evas_object_size_hint_min_set(label, ELM_SCALE_SIZE(480), 0);
    elm_label_line_wrap_set(label, ELM_WRAP_WORD);
    elm_object_text_set(label, "<font align=center>Outside events are blocked while notification shows.</center>");
    elm_box_pack_end(box, label);
    evas_object_show(label);

    elm_object_content_set(notify, box);
    return notify;
}
```

`elm_notify_allow_events_set(Evas_Object *, Eina_Bool)` 는 Notify가 Show 상태일때 사용자 이벤트를 허용할지 여부를 지정하는 함수입니다. 2번째 파라미터에 EINA_FALSE를 전달하면 사용자 이벤트가 차단됩니다.

예제를 다시 실행하고 3번째 Button을 눌러봅시다. Notify가 나타나고 3초 후에 다시 사라집니다. Notify가 Show 상태일 때는 다른 Button을 눌러도 반응이 없습니다.



5) Notify 위치 변경

Notify가 아래쪽에 표시되는 기능을 구현해 보겠습니다.
`create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Top / Block");
evas_object_smart_callback_add(btn, "clicked", btn_click_cb, notify);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
```

```

    notify = create_notify_bottom_timeout(box);

    /* Button-4 */
    btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "Bottom / Timeout");
    evas_object_smart_callback_add(btn, "clicked", btn_click_cb, notify);
    my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
}

}

```

4번째 notify와 4번째 Button을 생성하는 코드입니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```

static Evas_Object*
create_notify_bottom_timeout(Evas_Object *parent)
{
    Evas_Object *notify;
    Evas_Object *box;
    Evas_Object *label;

    /* Create notify (bottom-aligned / hide automatically) */
    notify = elm_notify_add(parent);
    elm_notify_align_set(notify, 0.5, 1.0);
    elm_notify_timeout_set(notify, 3.0);

    /* Create box for stacking notify message */
    box = elm_box_add(notify);
    evas_object_show(box);

    /* Create label for notify message */
    label = elm_label_add(box);
    evas_object_size_hint_min_set(label, ELM_SCALE_SIZE(480), 0);
    elm_label_line_wrap_set(label, ELM_WRAP_WORD);

```

```

elm_object_text_set(label, "<font align=center>This notification shows at the bottom
of the screen.</center>");
elm_box_pack_end(box, label);
evas_object_show(label);

elm_object_content_set(notify, box);
return notify;
}

```

`elm_notify_align_set(Evas_Object *, double, double)` 는 Notify의 위치를
비율 단위로 지정하는 API입니다. 2번째 파라미터는 수평 위치를
지정합니다. 0이면 왼쪽, 0.5면 가운데, 1이면 오른쪽에 배치됩니다. 3번째
파라미터는 수직 위치를 지정합니다. 0이면 위쪽, 0.5면 중앙, 1이면
아래쪽에 배치됩니다.

예제를 다시 실행하고 4번째 Button을 눌러봅시다. Notify가 화면
아래쪽에 나타났다가 잠시 후에 사라집니다.



6) 관련 API

Evas_Object *elm_notify_add(Evas_Object *parent) : Notify 객체를 생성하는 API.

void elm_notify_align_set(Evas_Object *obj, double horizontal, double vertical) : Notify의 위치를 비율 단위로 지정하는 API. 2번째 파라미터는 수평 위치를 지정합니다. 0이면 왼쪽, 0.5면 가운데, 1이면 오른쪽에 배치됩니다. 3번째 파라미터는 수직 위치를 지정합니다. 0이면 위쪽, 0.5면 중앙, 1이면 아래쪽에 배치됩니다.

void elm_notify_timeout_set(Evas_Object *obj, double timeout) : Notify에 Timeout을 지정하는 API. 2번째 파라미터에 시간 간격을 지정하며 단위는 초입니다.

void elm_notify_allow_events_set(Evas_Object *obj, Eina_Bool allow) : Notify가 Show 상태일때 사용자 이벤트를 허용할지 여부를 지정하는 함수. 2번째 파라미터에 EINA_FALSE를 전달하면 사용자 이벤트가 차단됩니다.

51. Acceleration 센서 사용방법

폰을 흔드는 동작은 가속도 센서로 측정할 수 있습니다. X, Y, Z 축 방향을 측정할 수 있으며 중력이 적용된 값과 적용되지 않는 결과를 구분해서 측정할 수 있습니다. 에뮬레이터에서 가속도 센서를 테스트 하려면 Control Panel을 사용하면 됩니다.

1) 가속도 센서 지원 여부 판단

새로운 소스 프로젝트를 생성하고 Project name을 SensorAcceleration으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "sensoracceleration.h"  
#include <sensor.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label0;  
    Evas_Object *label1;  
    Evas_Object *label2;  
} appdata_s;
```

sensor.h 는 각종 센서 라이브러리 헤더파일입니다.

label0에는 가속도 센서 지원 여부를 표시하고, label1에는 현재 가속도 값을 표시하고 label2에는 가속도 최대값을 표시하겠습니다.

`create_base_gui()` 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

`create_base_gui()` 함수 위에 새로운 함수 2개를 생성합니다.

```
static void show_is_supported(appdata_s *ad)
{
    char buf[PATH_MAX];
    bool is_supported = false;
    sensor_is_supported(SENSOR_ACCELEROMETER, &is_supported);
    sprintf(buf, "Acceleration Sensor is %s", is_supported ? "support" : "not support");
    elm_object_text_set(ad->label0, buf);
}

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
```

```

    /* actually put the child in the frame and show it */
    evas_object_show(child);
    elm_object_content_set(frame, child);
}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);

}

```

`show_is_supported()` 는 가속도 센서가 지원되는지를 판단해서 결과를 1번째 Label 위젯에 표시하는 함수입니다.

`sensor_is_supported(sensor_type_e, bool *)` 는 특정 센서의 지원여부를 구하는 API입니다. 1번째 파라미터에 `SENSOR_ACCELEROMETER`를 전달하면, 2번째 파라미터에서 센서 지원 여부를 반환해 줍니다.

`my_box_pack()` 은 Box에 위젯을 추가하는 함수입니다.

앱이 실행될 때 `show_is_supported()` 함수를 호출해 주면 됩니다. `create_base_gui()` 함수 끝부분에서 위 함수를 호출합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */

```

```

Evas_Object * box, *btn;

/* A box to put things in vertically - default mode for box */
box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
/* Label-0 */
ad->label0 = elm_label_add(ad->conform);
elm_object_text_set(ad->label0, "Msg - ");
my_box_pack(box, ad->label0, 1.0, 0.0, -1.0, 0.0);

/* Label-1 */
ad->label1 = elm_label_add(ad->conform);
elm_object_text_set(ad->label1, "Value - ");
my_box_pack(box, ad->label1, 1.0, 0.0, -1.0, 0.0);
}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);

}

```

2개의 Label 위젯을 추가하였습니다. 그리고 센서 지원여부 판단 함수를 호출했습니다.

예제를 빌드하고 실행시켜 봅시다. 가속도 센서가 지원되면 'Acceleration Sensor is support'라는 문구가 표시됩니다. 스마트 폰 중에서 센서가

지원되지 않는 경우도 있습니다. 그런 경우에는 에뮬레이터에서 테스트하기 바랍니다.

Acceleration Sensor is support

Value -

2) 가속도 센서 이벤트 구하기

디바이스를 흔들면 그 이벤트를 구해서 가속도 수치를 화면에 표시하는 기능을 구현해 보겠습니다. 소스파일 위쪽에 센서 관련 구조체와 전역변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label0;
    Evas_Object *label1;
    Evas_Object *label2;
} appdata_s;

typedef struct _sensor_info
{
    sensor_h sensor;      /**< Sensor handle */
    sensor_listener_h sensor_listener;
} sensorinfo;

static sensorinfo sensor_info;
```

sensorinfo 는 센서 객체와 이벤트 리스너 변수를 포함하고 있는 구조체입니다.

sensor_info 는 sensorinfo 구조체의 전역변수입니다.

센서 이벤트를 구하는 것은 리스너를 시작하는 것입니다. 센서 객체와 이벤트 리스너를 사용해서 가속도 센서 이벤트를 구해보겠습니다.
create_base_gui() 함수 위쪽에 새로운 함수 2개를 생성합니다.

```
static void _new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void *user_data)
{
    if( sensor_data->value_count < 3 )
        return;
    char buf[PATH_MAX];
    appdata_s *ad = (appdata_s*)user_data;

    sprintf(buf, "Value - X : %0.1f / Y : %0.1f / Z : %0.1f",
            sensor_data->values[0], sensor_data->values[1], sensor_data->values[2]);
    elm_object_text_set(ad->label1, buf);
}

static void
start_acceleration_sensor(appdata_s *ad)
{
    sensor_error_e err = SENSOR_ERROR_NONE;
    sensor_get_default_sensor(SENSOR_ACCELEROMETER, &sensor_info.sensor);
    err = sensor_create_listener(sensor_info.sensor, &sensor_info.sensor_listener);
    sensor_listener_set_event_cb(sensor_info.sensor_listener, 100, _new_sensor_value, ad);
    sensor_listener_start(sensor_info.sensor_listener);
}
```

_new_sensor_value() 는 가속도 센서 이벤트 콜백 함수입니다. 새로운 센서 값을 화면에 출력합니다. 2번째 파라미터에 센서 데이터가 배열로 전달됩니다. values[0]에는 x축 방향 데이터, values[1]에는 y축 방향

데이터, values[2]에는 z축 방향 데이터가 저장되어 있습니다.

`start_acceleration_sensor()` 는 가속도 센서를 시작하고, 이벤트 콜백 함수를 지정하는 함수입니다.

`sensor_get_default_sensor(sensor_type_e, sensor_h *)` 는 센서 객체를 반환하는 API입니다. 1번째 파라미터에 SENSOR_ACCELEROMETER을 전달하면, 2번째 파라미터에 가속도 센서 객체가 반환됩니다.

`sensor_create_listener(sensor_h, sensor_listener_h *)` 는 이벤트 리스너를 생성하는 API입니다. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

`sensor_listener_set_event_cb(sensor_listener_h, unsigned int, sensor_event_cb, void *)` 는 리스너에 콜백 함수를 지정하는 API입니다. 파라미터는 순서대로 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

`sensor_listener_start(sensor_listener_h)` 는 리스너를 시작하는 API 입니다.

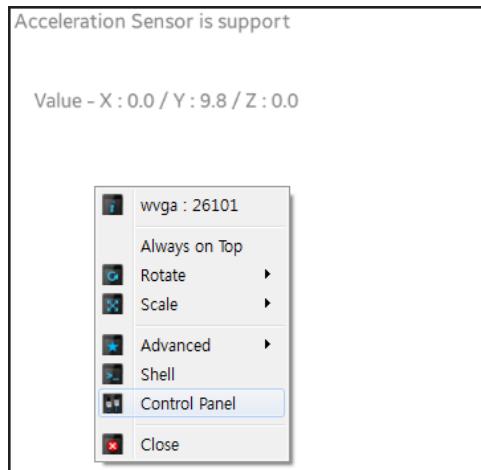
앱이 실행되면 자동으로 이벤트 리스너를 가동시켜 보겠습니다.
`create_base_gui()` 함수 끝부분에 위 함수를 호출해 줍니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);
start_acceleration_sensor(ad);
}
```

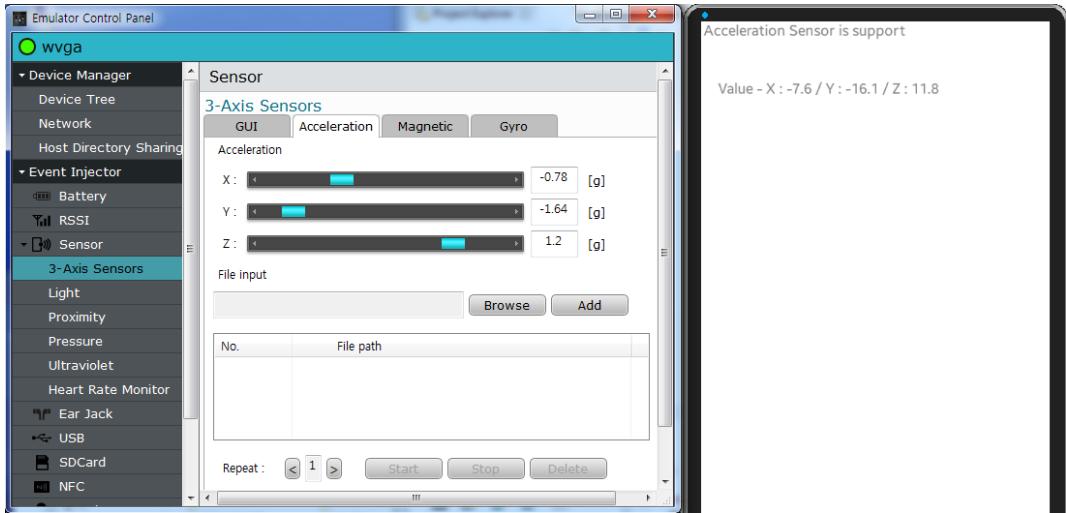
예제를 다시 실행시켜 보겠습니다. 스마트폰에서 테스트 할 때는 폰을 흔들면 됩니다. 에뮬레이터에서 테스트 할 때는 Control Panel을 사용하면 됩니다.

에뮬레이터를 마우스 오른쪽 버튼으로 클릭하고 단축 메뉴에서 Control Panel을 선택합니다.



Control Panel이 나타나면 왼쪽 트리 목록에서 [Event Injector > 3-Axis Sensors]를 선택하고, 그런 다음 오른쪽 화면 템버튼 중에서 Acceleration을 선택합니다.

3개의 슬라이더를 하나씩 드래그 해봅니다. 앱 화면에 X, Y, Z 수치가 변경되면 정상적으로 가속도 센서 데이터를 수신한 것입니다.



3) 가속도 최대값 구하기

스마트폰으로 테스트하게 되면 폰을 흔들때 글자가 제대로 보이지 않습니다. 그래서 수치를 보려고 흔드는걸 멈추면 아래쪽 방향은 9.8이고 나머지는 0으로 초기화 됩니다. 그래서 폰에서 테스트 할 때는 최대값을 저장하는 기능이 필요합니다.

소스파일 위쪽에 실수형 배열 변수를 선언하고 0으로 초기화 합니다.
가속도 최대값을 저장하는 변수입니다.

```

typedef struct _sensor_info
{
    sensor_h sensor;      /**< Sensor handle */
    sensor_listener_h sensor_listener;
} sensorinfo;

static sensorinfo sensor_info;

float value[3] = {0.f, 0.f, 0.f};

```

`create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
/* Label-1 */
ad->label1 = elm_label_add(ad->conform);
elm_object_text_set(ad->label1, "Value - ");
my_box_pack(box, ad->label1, 1.0, 0.0, -1.0, 0.0);

/* Button */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Init Max Value");
evas_object_smart_callback_add(btn, "clicked", btn_clicked, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, -1.0);

/* Label-2 */
ad->label2 = elm_label_add(ad->conform);
elm_object_text_set(ad->label2, "Max - ");
my_box_pack(box, ad->label2, 1.0, 1.0, -1.0, -1.0);

}
```

Button 위젯과 Label 위젯을 추가하는 코드입니다. 3번째 Label에는 가속도 센서 최대값을 표시하고, Button을 누르면 최대값을 0으로 초기화 하는 기능을 구현하겠습니다. 2개의 새로운 함수를 추가하고, `_new_sensor_value()` 함수에 새로운 코드를 추가합니다.

```
static float get_absolute_max(float value1, float value2)
{
    float v1 = value1 > 0.f ? value1 : -value1;
    float v2 = value2 > 0.f ? value2 : -value2;
    float result = v1 > v2 ? v1 : v2;
    return result;
```

```

}

static void _new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void
*user_data)
{
    if( sensor_data->value_count < 3 )
        return;
    char buf[PATH_MAX];
    appdata_s *ad = (appdata_s*)user_data;

    sprintf(buf, "Value - X : %0.1f / Y : %0.1f / Z : %0.1f",
            sensor_data->values[0], sensor_data->values[1], sensor_data->values[2]);
    elm_object_text_set(ad->label1, buf);

    for(int i=0; i < 3; i++)
        value[i] = get_absolute_max(value[i], sensor_data->values[i]);

    sprintf(buf, "Max - X: %0.1f / Y: %0.1f / Z: %0.1f",
            value[0], value[1], value[2]);
    elm_object_text_set(ad->label2, buf);
}

/* Button click event function */
static void
btn_clicked(void *data, Evas_Object *obj, void *event_info)
{
    for(int i=0; i < 3; i++)
        value[i] = 0.f;
}

```

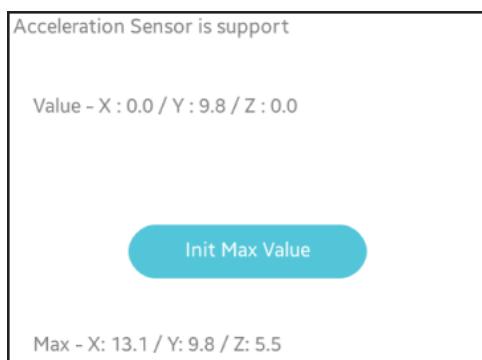
get_absolute_max(float, float) 는 2개의 실수값을 절대값으로 변경해서 더 큰값을 반환하는 함수입니다.

`_new_sensor_value()` 함수에 추가된 코드는 X,Y,Z 축의 최대값을 전역변수에 저장하고, Label 위젯에 출력하는 코드입니다.

`btn_clicked()` 는 사용자가 Button을 눌렀을 때 전역변수에 저장된 최대값을 0으로 초기화 하는 함수입니다.

예제를 폰에 설치하고 흔들어 봅시다. 멈추면 2번째 Label에 수치는 초기화 되지만, 3번째 Label에는 최대값이 그대로 남아있습니다.

Button을 누르고 다시 흔들면 새로운 값을 측정할 수 있습니다.



4) 중력을 제외한 순수 가속도 구하기

테스트 결과값을 보면 눈치채셨겠지만 우리가 구한 가속도에는 중력이 포함되어 있습니다. 중력이 제거된 순수한 가속도 값을 구해 보겠습니다.

`start_acceleration_sensor()` 함수의 코드를 아래와 같이 수정합니다.

```
static void
start_acceleration_sensor(appdata_s *ad)
{
    sensor_error_e err = SENSOR_ERROR_NONE;
```

```
//sensor_get_default_sensor(SENSOR_ACCELEROMETER, &sensor_info.sensor);
sensor_get_default_sensor(SENSOR_LINEAR_ACCELERATION, &sensor_info.sensor);
err = sensor_create_listener(sensor_info.sensor, &sensor_info.sensor_listener);
sensor_listener_set_event_cb(sensor_info.sensor_listener, 100, _new_sensor_value, ad);
sensor_listener_start(sensor_info.sensor_listener);
}
```

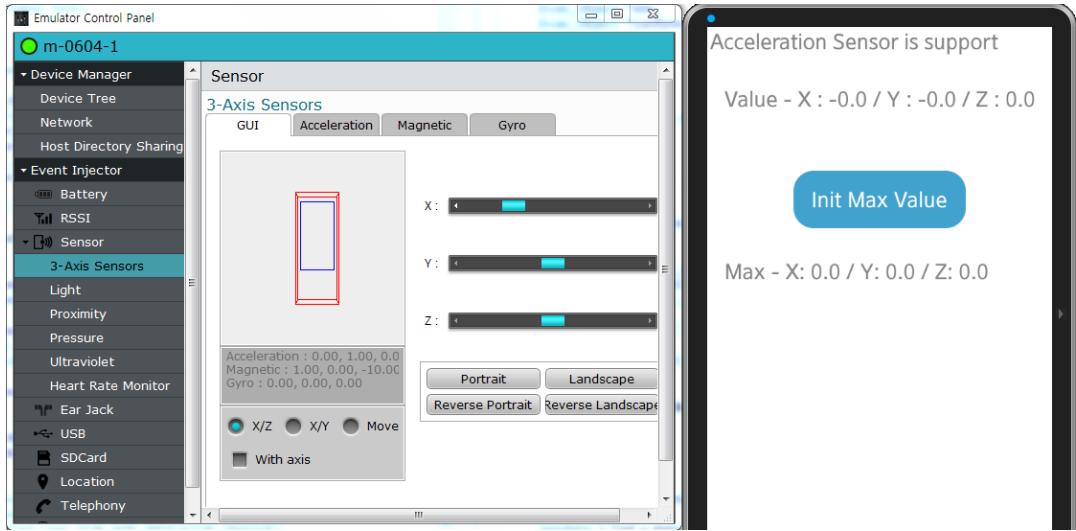
센서 종류 중에서 SENSOR_ACCELEROMETER 는 중력이 포함된 가속도를 의미합니다.

SENSOR_LINEAR_ACCELERATION 는 중력이 제거된 순수한 가속도를 의미합니다.

예제를 폰에 설치하고 흔들어 봅시다. 이제 순수한 가속도가 표시됩니다.

에뮬레이터에서 테스트 할 때는 Control Panel에서 [Event Injector > 3-Axis Sensors]를 선택하고 오른쪽 탭버튼에서 GUI를 클릭합니다.

Portrait 버튼과 Landscape 버튼을 번갈아서 눌러 봅시다. 센서 종류를 SENSOR_ACCELEROMETER 으로 지정했을 때는 X, Y, Z 의 총합이 9.80이 됩니다. SENSOR_LINEAR_ACCELERATION 으로 지정했을 때는 0이 됩니다.



5) 관련 API

`int sensor_is_supported(sensor_type_e type, bool *supported)` : 특정 센서의 지원여부를 구하는 API. 1번째 파라미터에 SENSOR_ACCELEROMETER를 전달하면, 2번째 파라미터에서 가속도 센서 지원 여부를 반환해 줍니다.

- SENSOR_ACCELEROMETER : 중력이 포함된 가속도 센서.
- SENSOR_LINEAR_ACCELERATION : 중력이 제거된 순수한 가속도 센서.

`int sensor_get_default_sensor(sensor_type_e type, sensor_h *sensor)` : 센서 객체를 반환하는 API. 1번째 파라미터에 SENSOR_ACCELEROMETER를 전달하면, 2번째 파라미터에 가속도 센서 객체가 반환됩니다.

`int sensor_create_listener(sensor_h sensor, sensor_listener_h *listener)` : 이벤트 리스너를 생성하는 API. 1번째 파라미터에 센서 객체를

전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

int sensor_listener_set_event_cb(sensor_listener_h listener, unsigned int interval_ms, sensor_event_cb callback, void *data) : 리스너에 콜백 함수를 지정하는 API. / 파라미터 : 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

int sensor_listener_start(sensor_listener_h listener) : 리스너를 시작하는 API.

52. Gravity 센서 사용방법

디바이스의 방향은 중력 센서로 측정할 수 있습니다. X, Y, Z 축 방향을 측정할 수 있습니다. 에뮬레이터에서 중력 센서를 테스트 하려면 Control Panel을 사용하면 됩니다.

1) 중력 센서 지원 여부 판단

새로운 소스 프로젝트를 생성하고 Project name을 SensorGravity 로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "sensorgravity.h"  
#include <sensor.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label0;  
    Evas_Object *label1;  
} appdata_s;
```

sensor.h는 각종 센서 라이브러리 헤더파일 입니다.

label0에는 중력 센서 지원 여부를 표시하고, label1에는 현재 중력 값을 표시하겠습니다.

create_base_gui() 함수 위에 2개의 새로운 함수를 생성합니다.

```

static void show_is_supported(appdata_s *ad)
{
    char buf[PATH_MAX];
    bool is_supported = false;
    sensor_is_supported(SENSOR_GRAVITY, &is_supported);
    sprintf(buf, "Gravity Sensor is %s", is_supported ? "support" : "not support");
    elm_object_text_set(ad->label0, buf);
}

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
}

```

```

/* show the frame */
evas_object_show(frame);
}

```

`show_is_supported()` 는 중력 센서가 지원되는지를 판단해서 결과를 1번째 Label 위젯에 표시하는 함수입니다.

`sensor_is_supported(sensor_type_e, bool *)` 는 특정 센서의 지원여부를 구하는 API입니다. 1번째 파라미터에 `SENSOR_GRAVITY`를 전달하면, 2번째 파라미터에서 센서 지원 여부를 반환해 줍니다.

`my_box_pack()` 은 Box에 위젯을 추가하는 함수입니다.

`show_is_supported()` 함수를 앱이 실행될 때 호출해 주면 됩니다. `create_base_gui()` 함수 끝부분에서 위 함수를 호출합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
Evas_Object * box, *btn;

/* A box to put things in vertically - default mode for box */
box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

```

```

elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
    /* Label-0 */
    ad->label0 = elm_label_add(ad->conform);
    elm_object_text_set(ad->label0, "Msg - ");
    my_box_pack(box, ad->label0, 1.0, 0.0, -1.0, 0.0);

    /* Label-1 */
    ad->label1 = elm_label_add(ad->conform);
    elm_object_text_set(ad->label1, "Value - ");
    my_box_pack(box, ad->label1, 1.0, 1.0, -1.0, 0.0);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);
}

```

Box 컨테이너와 2개의 Label 위젯을 추가하였습니다. 그리고 센서 지원여부 판단 함수를 호출했습니다.

예제를 빌드하고 실행시켜 봅시다. 중력 센서가 지원되면 'Gravity Sensor is support'라는 문구가 표시됩니다. 스마트 폰 중에서 센서가 지원되지 않는 경우도 있습니다. 그런 경우에는 에뮬레이터에서 테스트하기 바랍니다.



2) 중력 센서 이벤트 구하기

디바이스 방향이 바뀌면 그 이벤트를 구해서 중력 수치를 화면에 표시하는 기능을 구현해 보겠습니다. 소스파일 위쪽에 센서 관련 구조체와 전역변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label0;
    Evas_Object *label1;
} appdata_s;

typedef struct _sensor_info
{
    sensor_h sensor;      /**< Sensor handle */
    sensor_listener_h sensor_listener;
} sensorinfo;

static sensorinfo sensor_info;
```

sensorinfo는 센서 객체와 이벤트 리스너 변수를 포함하고 있는 구조체입니다.

sensor_info는 sensorinfo 구조체의 전역변수입니다.

센서 이벤트를 구하는 것은 리스너를 시작하는 것입니다. 센서 객체와 이벤트 리스너를 사용해서 중력 센서 이벤트를 구해보겠습니다. create_base_gui() 함수 위쪽에 새로운 함수 2개를 생성합니다.

```

static void _new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void
*user_data)
{
    if( sensor_data->value_count < 3 )
        return;
    char buf[PATH_MAX];
    appdata_s *ad = (appdata_s*)user_data;

    sprintf(buf, "Gravity - X : %0.1f / Y : %0.1f / Z : %0.1f",
            sensor_data->values[0], sensor_data->values[1], sensor_data->values[2]);
    elm_object_text_set(ad->label1, buf);
}

static void
start_gravity_sensor(appdata_s *ad)
{
    sensor_error_e err = SENSOR_ERROR_NONE;
    sensor_get_default_sensor(SENSOR_GRAVITY, &sensor_info.sensor);
    err = sensor_create_listener(sensor_info.sensor, &sensor_info.sensor_listener);
    sensor_listener_set_event_cb(sensor_info.sensor_listener, 100, _new_sensor_value, ad);
    sensor_listener_start(sensor_info.sensor_listener);
}

```

`_new_sensor_value()` 는 중력 센서 이벤트 콜백 함수입니다. 새로운 센서 값을 화면에 출력합니다.

2번째 파라미터에 센서 데이터가 배열로 전달됩니다. `values[0]`에는 x축 방향 데이터, `values[1]`에는 y축 방향 데이터, `values[2]`에는 z축 방향 데이터가 저장되어 있습니다.

`start_gravity_sensor()` 는 중력 센서를 시작하고, 이벤트 콜백 함수를 지정하는 함수입니다.

`sensor_get_default_sensor(sensor_type_e, sensor_h *)` 는 센서 객체를 반환하는 API입니다. 1번째 파라미터에 SENSOR_GRAVITY 를 전달하면, 2번째 파라미터에 중력 센서 객체가 반환됩니다.

`sensor_create_listener(sensor_h, sensor_listener_h *)` 는 이벤트 리스너를 생성하는 API입니다. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

`sensor_listener_set_event_cb(sensor_listener_h, unsigned int, sensor_event_cb, void *)` 는 리스너에 콜백 함수를 지정하는 API입니다. 파라미터는 순서대로 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

`sensor_listener_start(sensor_listener_h)` 는 리스너를 시작하는 API 입니다.

앱이 실행되면 자동으로 이벤트 리스너를 가동시켜 보겠습니다.
`create_base_gui()` 함수 끝부분에 위 함수를 호출해 줍니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);
start_gravity_sensor(ad);

}
```

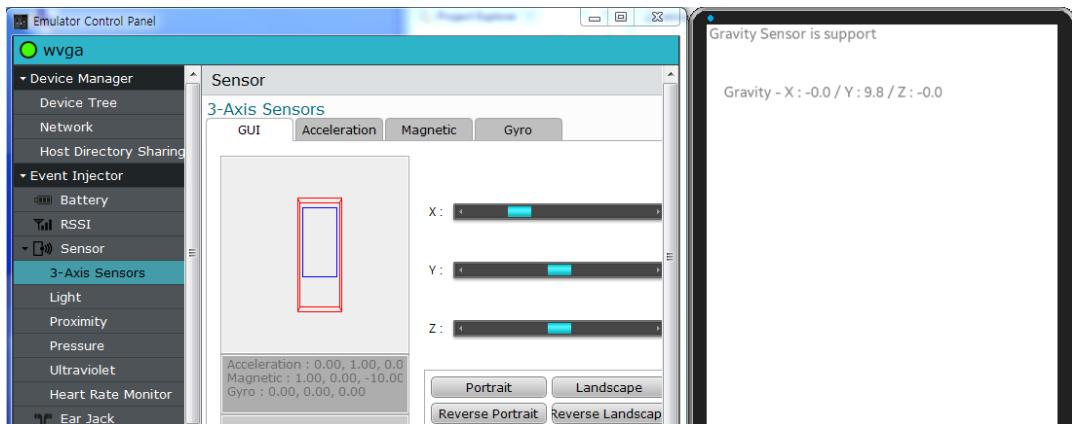
예제를 다시 실행시켜 보겠습니다. 스마트폰에서 테스트 할 때는 폰을 회전하면 됩니다. 에뮬레이터에서 테스트 할 때는 Control Panel을 사용하면 됩니다.

에뮬레이터를 마우스 오른쪽 버튼으로 클릭하고 단축 메뉴에서 Control

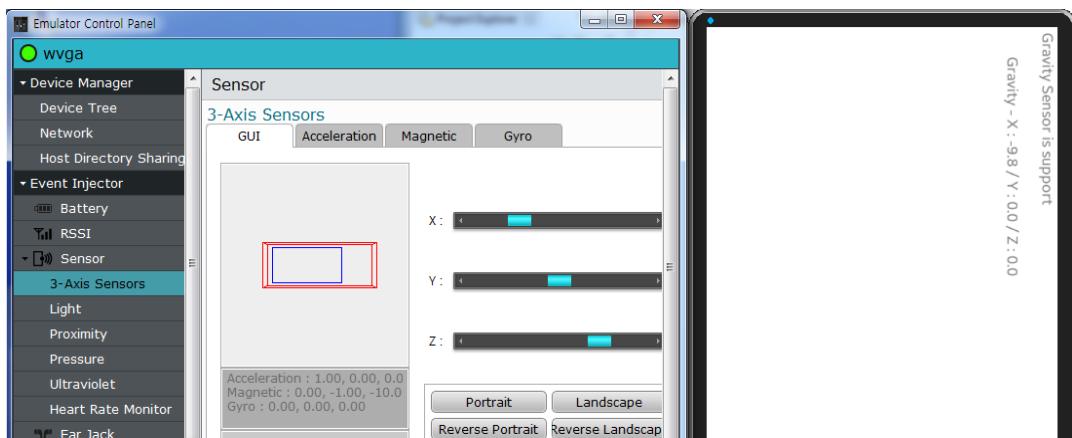
Panel을 선택합니다.

Control Panel 이 나타나면 왼쪽 트리 목록에서 [Event Injector > 3-Axis Sensors]를 선택하고, 그런 다음 오른쪽 화면 템버튼 중에서 GUI를 선택합니다.

Control Panel 오른쪽 화면에서 Portrait 버튼을 누르면 앱 화면의 2번째 Label 위젯에 'X : 0.0 / Y : 9.8 / Z : 0.0' 이 표시됩니다.



Control Panel 오른쪽 화면에서 Landscape 버튼을 누르면 앱 화면의 2번째 Label 위젯에 'X : 9.8 / Y : 0.0 / Z : 0.0' 이 표시됩니다.



3) 관련 API

int sensor_is_supported(sensor_type_e type, bool *supported) : 특정 센서의 지원여부를 구하는 API. 1번째 파라미터에 SENSOR_GRAVITY를 전달하면, 2번째 파라미터에서 중력 센서 지원 여부를 반환해 줍니다.

int sensor_get_default_sensor(sensor_type_e type, sensor_h *sensor) : 센서 객체를 반환하는 API. 1번째 파라미터에 SENSOR_GRAVITY를 전달하면, 2번째 파라미터에 중력 센서 객체가 반환됩니다.

int sensor_create_listener(sensor_h sensor, sensor_listener_h *listener) : 이벤트 리스너를 생성하는 API. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

int sensor_listener_set_event_cb(sensor_listener_h listener, unsigned int interval_ms, sensor_event_cb callback, void *data) : 리스너에 콜백 함수를 지정하는 API. / 파라미터 : 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

int sensor_listener_start(sensor_listener_h listener) : 리스너를 시작하는 API.

53. Orientation 센서 사용방법

Orientation 센서는 3가지 종류의 방향을 측정할 수 있습니다.

- Azimuth는 나침반 센서입니다. 폰을 바닥에 누웠을 때 북극 방향과의 각도 차이를 알려줍니다.
- Pitch는 폰을 세웠을 때 Z축 방향의 각도를 알려줍니다. 주로 비행기 게임 혹은 자동차 게임의 핸들 역할을 합니다.
- Roll은 Landscape 모드로 폰을 눕혔을 때 Y축 방향의 각도를 알려줍니다. 주로 비행기 게임 혹은 자동차 게임의 속도 제어 역할을 합니다.

폰을 Portrait 모드로 세웠을 때 수평 방향이 X축이고, 수직 방향이 Y축입니다. 그리고 앞뒤 방향이 Z축입니다. 에뮬레이터에서 Orientation 센서를 테스트 하려면 Control Panel을 사용하면 됩니다.

Figure: Axis of the device



1) Orientation 센서 지원 여부 판단

새로운 소스 프로젝트를 생성하고 Project name을 SensorOrientation 로

지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "sensororientation.h"
#include <sensor.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label0;
    Evas_Object *label1;
} appdata_s;
```

sensor.h는 각종 센서 라이브러리 헤더파일입니다.

label0에는 Orientation 센서 지원 여부를 표시하고, label1에는 현재 Orientation 값을 표시하겠습니다.

create_base_gui() 함수 위에 2개의 새로운 함수를 생성합니다.

```
static void
show_is_supported(appdata_s *ad)
{
    char buf[PATH_MAX];
    bool is_supported = false;
    sensor_is_supported(SENSOR_ORIENTATION, &is_supported);
    sprintf(buf, "Orientation Sensor is %s", is_supported ? "support" : "not support");
    elm_object_text_set(ad->label0, buf);
}

static void
```

```

my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}

```

show_is_supported() 는 Orientation 센서가 지원되는지를 판단해서 결과를 1번째 Label 위젯에 표시하는 함수입니다.

sensor_is_supported(sensor_type_e, bool *) 는 특정 센서의 지원여부를 구하는 API입니다. 1번째 파라미터에 SENSOR_ORIENTATION를 전달하면,

2번째 파라미터에서 센서 지원 여부를 반환해 줍니다.

my_box_pack() 은 Box에 위젯을 추가하는 함수입니다.

show_is_supported() 함수를 앱이 실행될 때 호출해 주면 됩니다.
create_base_gui() 함수 끝부분에서 위 함수를 호출합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
Evas_Object * box, *btn;

/* A box to put things in vertically - default mode for box */
box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
/* Label-0 */
ad->label0 = elm_label_add(ad->conform);
elm_object_text_set(ad->label0, "Msg - ");
my_box_pack(box, ad->label0, 1.0, 0.0, -1.0, 0.0);

/* Label-1 */
```

```

        ad->label1 = elm_label_add(ad->conform);
        elm_object_text_set(ad->label1, "Orientation - ");
        my_box_pack(box, ad->label1, 1.0, 1.0, -1.0, 0.0);

    }

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);

}

```

Box 컨테이너와 2개의 Label 위젯을 생성하였습니다. 그리고 센서 지원여부 판단 함수를 호출했습니다.

예제를 빌드하고 실행시켜 봅시다. Orientation 센서가 지원되면 'Orientation Sensor is support'라는 문구가 표시됩니다. 스마트 폰 중에서 센서가 지원되지 않는 경우도 있습니다. 그런 경우에는 에뮬레이터에서 테스트하기 바랍니다.



2) Orientation 센서 이벤트 구하기

디바이스 방향이 바뀌면 그 이벤트를 구해서 수치를 화면에 표시하는 기능을 구현해 보겠습니다. 소스파일 위쪽에 센서 관련 구조체와 전역변수를 추가합니다.

```

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label0;
    Evas_Object *label1;
} appdata_s;

typedef struct _sensor_info
{
    sensor_h sensor;      /*< Sensor handle */
    sensor_listener_h sensor_listener;
} sensorinfo;

static sensorinfo sensor_info;

```

sensorinfo는 센서 객체와 이벤트 리스너 변수를 포함하고 있는 구조체입니다.

sensor_info는 sensorinfo 구조체의 전역변수 입니다.

센서 이벤트를 구하는 것은 리스너를 시작하는 것입니다. 센서 객체와 이벤트 리스너를 사용해서 Orientation 센서 이벤트를 구해보겠습니다. create_base_gui() 함수 위쪽에 새로운 함수 2개를 생성합니다.

```

static void
_new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void *user_data)
{
    if( sensor_data->value_count < 3 )
        return;
    char buf[PATH_MAX];
    appdata_s *ad = (appdata_s*)user_data;

```

```

        sprintf(buf, "Azimuth : %0.1f <br>Pitch : %0.1f <br>Roll : %0.1f",
            sensor_data->values[0], sensor_data->values[1], sensor_data->values[2]);
        elm_object_text_set(ad->label1, buf);
    }

static void
start_orientation_sensor(appdata_s *ad)
{
    sensor_error_e err = SENSOR_ERROR_NONE;
    sensor_get_default_sensor(SENSOR_ORIENTATION, &sensor_info.sensor);
    err = sensor_create_listener(sensor_info.sensor, &sensor_info.sensor_listener);
    sensor_listener_set_event_cb(sensor_info.sensor_listener, 100, _new_sensor_value, ad);
    sensor_listener_start(sensor_info.sensor_listener);
}

```

`_new_sensor_value()` 는 Orientation 센서 이벤트 콜백 함수입니다.
새로운 센서 값을 화면에 출력합니다.

2번째 파라미터에 센서 데이터가 배열로 전달됩니다. `values[0]`에는 Azimuth 데이터, `values[1]`에는 Pitch 데이터, `values[2]`에는 Roll 데이터가 저장되어 있습니다.

`start_orientation_sensor()` 는 Orientation 센서를 시작하고, 이벤트 콜백 함수를 지정하는 함수입니다.

`sensor_get_default_sensor(sensor_type_e, sensor_h *)` 는 센서 객체를 반환하는 API입니다. 1번째 파라미터에 `SENSOR_ORIENTATION` 를 전달하면, 2번째 파라미터에 Orientation 센서 객체가 반환됩니다.

`sensor_create_listener(sensor_h, sensor_listener_h *)` 는 이벤트 리스너를 생성하는 API입니다. 1번째 파라미터에 센서 객체를 전달하면, 2번째

파라미터에 리스너 객체가 반환됩니다.

sensor_listener_set_event_cb(sensor_listener_h, unsigned int, sensor_event_cb, void *) 는 리스너에 콜백 함수를 지정하는 API입니다. 파라미터는 순서대로 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

sensor_listener_start(sensor_listener_h) 는 리스너를 시작하는 API 입니다.

앱이 실행되면 자동으로 이벤트 리스너를 가동시켜 보겠습니다.

create_base_gui() 함수 끝부분에 위 함수를 호출해 줍니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);
start_orientation_sensor(ad);

}
```

이 상태로 테스트하면 폰을 회전했을 때 Landscape 모드로 변경될 수 있습니다. 화면 방향을 Portrait 모드로 고정 시키겠습니다.

create_base_gui() 함수 위부분에 아래 코드는 4가지 방향을 지원하게 해줍니다.

```
int rots[4] = { 0, 90, 180, 270 };
elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots), 4);
```

이 코드를 다음과 같이 수정합니다.

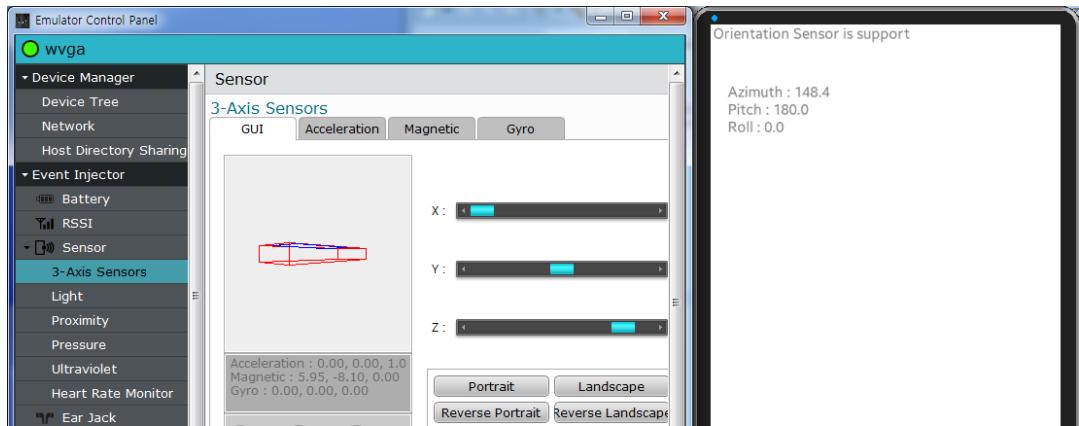
```
int rots[1] = { 0 };
elm_win_wm_rotation_available_rotations_set(ad->win, (const int *)(&rots), 1);
```

예제를 다시 실행시켜 보겠습니다. 스마트폰에서 테스트 할 때는 폰을 회전하면 됩니다. 에뮬레이터에서 테스트 할 때는 Control Panel을 사용하면 됩니다.

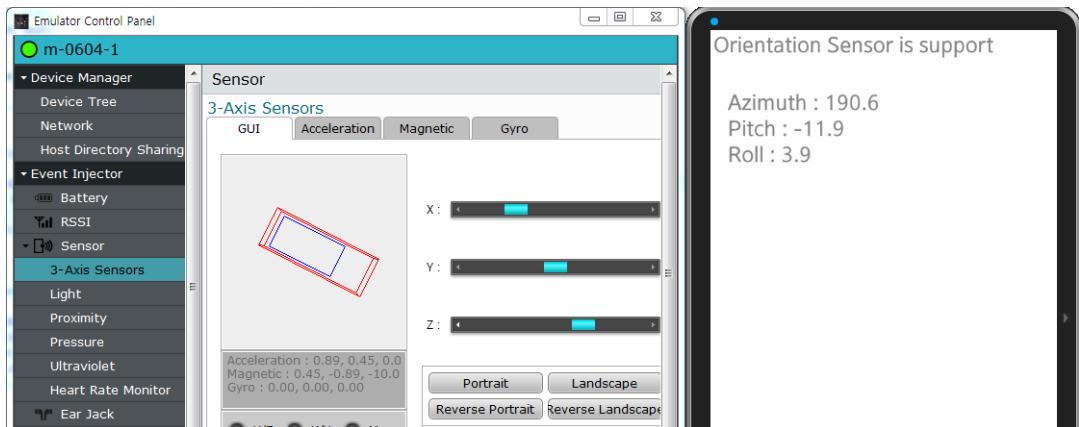
에뮬레이터를 마우스 오른쪽 버튼으로 클릭하고 단축 메뉴에서 Control Panel을 선택합니다.

Control Panel이 나타나면 왼쪽 트리 목록에서 [Event Injector > 3-Axis Sensors]를 선택하고, 그런 다음 오른쪽 화면 탭버튼 중에서 GUI를 선택합니다.

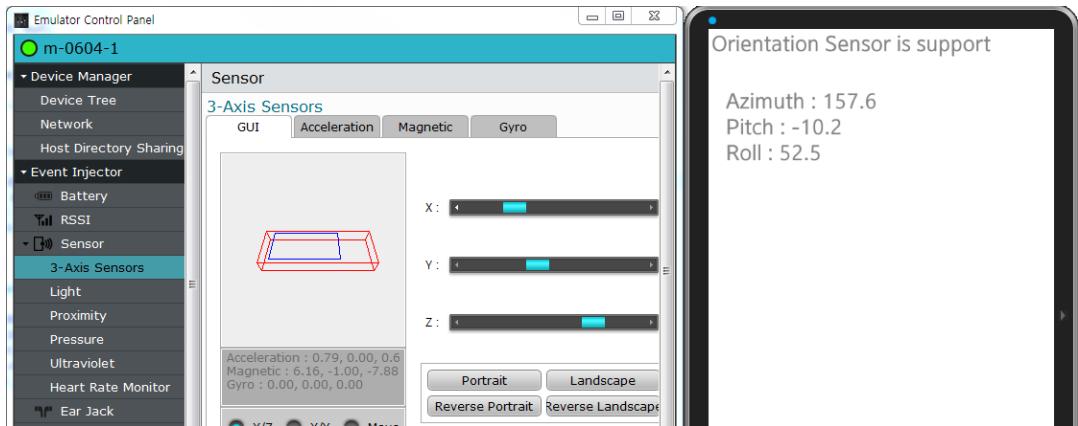
1번째로 Azimuth를 테스트 해보겠습니다. Portrait 버튼을 누르고, X축 슬라이더를 왼쪽 끝으로 이동시키면 폰이 바닥에 눕게 됩니다. 이 상태에서 Z축 슬라이더를 좌우로 드래그 해봅니다. 앱화면에 Azimuth 수치가 0~360 까지 변합니다.



2번째로 Pitch를 테스트 해보겠습니다. Portrait 버튼을 누르고, 이 상태에서 Z축 슬라이더를 좌우로 드래그 해봅니다. 앱화면에 Azimuth 수치가 -180~180 까지 변합니다.



3번째로 Roll을 테스트 해보겠습니다. Landscape 버튼을 누르고, 이 상태에서 Y축 슬라이더를 좌우로 드래그 해봅니다. 앱화면에 Roll 수치가 -180~180 까지 변합니다.



3) 관련 API

`int sensor_is_supported(sensor_type_e type, bool *supported)` : 특정 센서의 지원여부를 구하는 API. 1번째 파라미터에 SENSOR_ORIENTATION를 전달하면, 2번째 파라미터에서 Orientation 센서 지원 여부를 반환해 줍니다.

`int sensor_get_default_sensor(sensor_type_e type, sensor_h *sensor)` : 센서 객체를 반환하는 API. 1번째 파라미터에 SSENSOR_ORIENTATION를 전달하면, 2번째 파라미터에 Orientation 센서 객체가 반환됩니다.

`int sensor_create_listener(sensor_h sensor, sensor_listener_h *listener)` : 이벤트 리스너를 생성하는 API. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

`int sensor_listener_set_event_cb(sensor_listener_h listener, unsigned int interval_ms, sensor_event_cb callback, void *data)` : 리스너에 콜백 함수를 지정하는 API. / 파라미터 : 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

`int sensor_listener_start(sensor_listener_h listener)` : 리스너를 시작하는 API.

54. 자기장 센서 사용방법

나침반 앱을 제작하거나 주변에 자기장의 세기를 측정하면 자기장 센서를 사용하면 됩니다. X,Y,Z 3가지 축의 세기로 측정할 수 있습니다. 폰을 Portrait 모드로 세웠을 때 수평 방향이 X축이고, 수직 방향이 Y축입니다. 그리고 앞뒤 방향이 Z축입니다. 에뮬레이터에서 자기장 센서를 테스트 하려면 Control Panel을 사용하면 됩니다.

1) 자기장 센서 지원 여부 판단

새로운 소스 프로젝트를 생성하고 Project name을 SensorMagnetic 로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "sensormagnetic.h"  
#include <sensor.h>  
#include <math.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label0;  
    Evas_Object *label1;  
    Evas_Object *label2;  
} appdata_s;
```

sensor.h는 각종 센서 라이브러리 헤더파일입니다.

math.h는 수학 라이브러리 헤더파일입니다.

label0에는 자기장 센서 지원 여부를 표시하고, label1에는 3가지 축의 자기장 값을 표시하겠습니다. label2에는 전체 자기장 값을 표시하겠습니다.

create_base_gui() 함수 위에 2개의 새로운 함수를 생성합니다.

```
static void show_is_supported(appdata_s *ad)
{
    char buf[PATH_MAX];
    bool is_supported = false;
    sensor_is_supported(SENSOR_MAGNETIC, &is_supported);
    sprintf(buf, "Magnetic Sensor is %s", is_supported ? "support" : "not support");
    elm_object_text_set(ad->label0, buf);
}

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
}
```

```

/* tell the child that is packed into the frame to be able to expand */
evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);

/* fill the expanded area (above) as opposed to center in it */
evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
/* actually put the child in the frame and show it */
evas_object_show(child);
elm_object_content_set(frame, child);

}

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);

}

```

`show_is_supported()` 는 자기장 센서가 지원되는지를 판단해서 결과를 1번째 Label 위젯에 표시하는 함수입니다.

`sensor_is_supported(sensor_type_e, bool *)` 는 특정 센서의 지원여부를 구하는 API입니다. 1번째 파라미터에 `SENSOR_MAGNETIC` 을 전달하면, 2번째 파라미터에서 센서 지원 여부를 반환해 줍니다.

`my_box_pack()` 은 Box에 위젯을 추가하는 함수입니다.

위 함수를 앱이 실행될 때 호출해 주면 됩니다. `create_base_gui()` 함수 끝부분에서 위 함수를 호출합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
```

```

EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
  Evas_Object * box, *btn;

  /* A box to put things in vertically - default mode for box */
  box = elm_box_add(ad->win);
  evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
  elm_object_content_set(ad->conform, box);
  evas_object_show(box);

  { /* child object - indent to how relationship */
    /* Label-0 */
    ad->label0 = elm_label_add(ad->conform);
    elm_object_text_set(ad->label0, "Msg - ");
    my_box_pack(box, ad->label0, 1.0, 0.0, -1.0, 0.0);

    /* Label-1 */
    ad->label1 = elm_label_add(ad->conform);
    elm_object_text_set(ad->label1, "Value - ");
    my_box_pack(box, ad->label1, 1.0, 0.0, -1.0, 0.0);

    /* Label-2 */
    ad->label2 = elm_label_add(ad->conform);
    elm_object_text_set(ad->label2, "Strength : ");
    my_box_pack(box, ad->label2, 1.0, 1.0, -1.0, 0.0);
  }
}

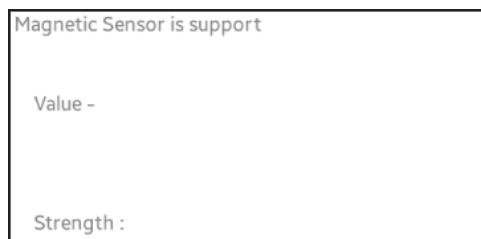
/* Show window after base gui is set up */
evas_object_show(ad->win);

```

```
    show_is_supported(ad);  
}  
_____
```

3개의 Label 위젯을 생성하였습니다. 그리고 센서 지원여부 판단 함수를 호출했습니다.

예제를 빌드하고 실행시켜 봅시다. 자기장 센서가 지원되면 'Magnetic Sensor is support'라는 문구가 표시됩니다. 스마트 폰 중에서 센서가 지원되지 않는 경우도 있습니다. 그런 경우에는 에뮬레이터에서 테스트하기 바랍니다.



2) 자기장 센서 이벤트 구하기

주변 자기장이 바뀌면 그 이벤트를 구해서 수치를 화면에 표시하는 기능을 구현해 보겠습니다. 소스파일 위쪽에 센서 관련 구조체와 전역변수를 추가합니다.

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label0;  
    Evas_Object *label1;  
    Evas_Object *label2;
```

```

} appdata_s;

typedef struct _sensor_info
{
    sensor_h sensor;      /*< Sensor handle */
    sensor_listener_h sensor_listener;
} sensorinfo;

static sensorinfo sensor_info;

```

sensorinfo는 센서 객체와 이벤트 리스너 변수를 포함하고 있는 구조체입니다.

sensor_info는 sensorinfo 구조체의 전역변수입니다.

센서 이벤트를 구하는 것은 리스너를 시작하는 것입니다. 센서 객체와 이벤트 리스너를 사용해서 자기장 센서 이벤트를 구해보겠습니다. create_base_gui() 함수 위쪽에 새로운 함수 2개를 생성합니다.

```

static void
_new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void *user_data)
{
    if( sensor_data->value_count < 3 )
        return;
    char buf[PATH_MAX];
    appdata_s *ad = (appdata_s*)user_data;

    sprintf(buf, "X : %0.1f / Y : %0.1f / Z : %0.1f",
            sensor_data->values[0], sensor_data->values[1], sensor_data->values[2]);
    elm_object_text_set(ad->label1, buf);
}

```

```
static void
start_magnetic_sensor(appdata_s *ad)
{
    sensor_error_e err = SENSOR_ERROR_NONE;
    sensor_get_default_sensor(SENSOR_MAGNETIC, &sensor_info.sensor);
    err = sensor_create_listener(sensor_info.sensor, &sensor_info.sensor_listener);
    sensor_listener_set_event_cb(sensor_info.sensor_listener, 100, _new_sensor_value, ad);
    sensor_listener_start(sensor_info.sensor_listener);
}
```

_new_sensor_value() 는 자기장 센서 이벤트 콜백 함수입니다. 새로운 센서 값을 화면에 출력합니다.

2번째 파라미터에 센서 데이터가 배열로 전달됩니다. values[0]에는 x축 방향 데이터, values[1]에는 y축 방향 데이터, values[2]에는 z축 방향 데이터가 저장되어 있습니다.

start_magnetic_sensor() 는 자기장 센서를 시작하고, 이벤트 콜백 함수를 지정하는 함수입니다.

sensor_get_default_sensor(sensor_type_e, sensor_h *) 는 센서 객체를 반환하는 API입니다. 1번째 파라미터에 SENSOR_MAGNETIC 를 전달하면, 2번째 파라미터에 자기장 센서 객체가 반환됩니다.

sensor_create_listener(sensor_h, sensor_listener_h *) 는 이벤트 리스너를 생성하는 API입니다. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

sensor_listener_set_event_cb(sensor_listener_h, unsigned int, sensor_event_cb, void *) 는 리스너에 콜백 함수를 지정하는 API입니다. 파라미터는 순서대로 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백

함수명, 사용자 데이터.

sensor_listener_start(sensor_listener_h) 는 리스너를 시작하는 API 입니다.

앱이 실행되면 자동으로 이벤트 리스너를 가동시켜 보겠습니다.
create_base_gui() 함수 끝부분에 위 함수를 호출해 줍니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);
start_magnetic_sensor(ad);

}
```

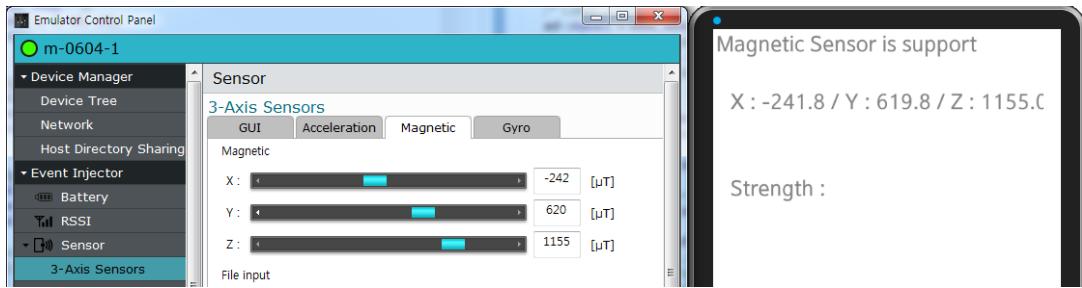
예제를 다시 실행시켜 보겠습니다. 스마트폰에서 테스트 할 때는 자석을
가까이 가져가거나 폰을 눕혀놓은 상태로 회전하면 됩니다.

에뮬레이터에서 테스트 할 때는 Control Panel을 사용하면 됩니다.

에뮬레이터를 마우스 오른쪽 버튼으로 클릭하고 단축 메뉴에서 Control
Panel을 선택합니다.

Control Panel 이 나타나면 왼쪽 트리 목록에서 [Event Injector > 3-Axis
Sensors]를 선택하고, 그런 다음 오른쪽 화면 탭버튼 중에서 Magnetic을
선택합니다.

Control Panel 오른쪽 화면에서 3가지 슬라이더를 드래그하면 앱 화면의
2번째 Label 위젯에 수치가 변경됩니다.



3) 전체 자기장 수치 구하기

X,Y,Z 축의 값을 종합한 전체 자기장 수치를 구하는 방법을 알아보았습니다. 전체 자기장 수치를 구하려면 3가지 값의 제곱을 더해서 제곱근을 구하면 됩니다. `_new_sensor_value()` 함수 위에 새로운 함수를 추가하고 `_new_sensor_value()` 함수의 코드를 수정합니다.

```

static float
_magagnetic_strength_get(const float *values)
{
    float sum = 0.0;
    for(int i=0; i < 3; i++)
        sum += values[i] * values[i];

    return sqrt(sum);
}

static void
_new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void *user_data)
{
    if( sensor_data->value_count < 3 )
        return;

    char buf[PATH_MAX];
    appdata_s *ad = (appdata_s*)user_data;
}

```

```

sprintf(buf, "X : %0.1f / Y : %0.1f / Z : %0.1f",
       sensor_data->values[0], sensor_data->values[1], sensor_data->values[2]);
elm_object_text_set(ad->label1, buf);

float strength = _magnetic_strength_get(sensor_data->values);
sprintf(buf, "Strength : %0.1f", strength);
elm_object_text_set(ad->label2, buf);

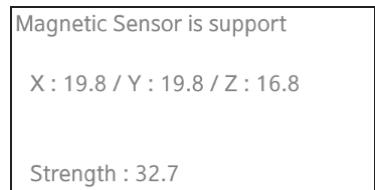
}

```

`_magnetic_strength_get(const float *)` 는 배열에 저장된 3가지 수치의 제곱을 모두 더해서 그 결과의 제곱근을 반환하는 함수입니다.

`sqrt(double)` 는 제곱근을 구하는 수학 API입니다.

예제를 다시 실행시켜 봅시다. 3번째 Label에 전체 자기장 수치 계산 결과가 표시됩니다.



4) 관련 API

`int sensor_is_supported(sensor_type_e type, bool *supported)` : 특정 센서의 지원여부를 구하는 API. 1번째 파라미터에 SSENSOR_MAGNETIC 을 전달하면, 2번째 파라미터에서 자기장 센서 지원 여부를 반환해 줍니다.

`int sensor_get_default_sensor(sensor_type_e type, sensor_h *sensor)` : 센서 객체를 반환하는 API. 1번째 파라미터에 SENSOR_MAGNETIC 을 전달하면, 2번째 파라미터에 자기장 센서 객체가 반환됩니다.

`int sensor_create_listener(sensor_h sensor, sensor_listener_h *listener)` : 이벤트 리스너를 생성하는 API. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

`int sensor_listener_set_event_cb(sensor_listener_h listener, unsigned int interval_ms, sensor_event_cb callback, void *data)` : 리스너에 콜백 함수를 지정하는 API. / 파라미터 : 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

`int sensor_listener_start(sensor_listener_h listener)` : 리스너를 시작하는 API.

`double sqrt(double)` : 제곱근을 구하는 수학 API.

55. Proximity 센서 사용방법

전화가 걸려왔을 때 얼굴을 폰에 가까이 대면 폰 화면이 자동으로 꺼집니다. 이것은 Proximity 센서를 사용한 것입니다. 에뮬레이터에서 Proximity 센서를 테스트 하려면 Control Panel을 사용하면 됩니다.

1) Proximity 센서 지원 여부 판단

새로운 소스 프로젝트를 생성하고 Project name을 SensorProximity로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "sensorproximity.h"  
#include <sensor.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label0;  
    Evas_Object *label1;  
} appdata_s;
```

sensor.h 는 각종 센서 라이브러리 헤더파일입니다.

label0에는 Proximity 센서 지원 여부를 표시하고, label1에는 거리 수치를 표시하겠습니다.

create_base_gui() 함수 위에 2개의 새로운 함수를 생성합니다.

```

static void
show_is_supported(appdata_s *ad)
{
    char buf[PATH_MAX];
    bool is_supported = false;
    sensor_is_supported(SENSOR_PROXIMITY, &is_supported);
    sprintf(buf, "Proximity Sensor is %s", is_supported ? "support" : "not support");
    elm_object_text_set(ad->label0, buf);
}

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */

```

```

elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}

```

`show_is_supported()` 는 Proximity 센서가 지원되는지를 판단해서 결과를 1번째 Label 위젯에 표시하는 함수입니다.

`sensor_is_supported(sensor_type_e, bool *)` 는 특정 센서의 지원여부를 구하는 API입니다. 1번째 파라미터에 SENSOR_PROXIMITY를 전달하면, 2번째 파라미터에서 센서 지원 여부를 반환해 줍니다.

`my_box_pack()`는 Box에 위젯을 추가하는 함수입니다.

`show_is_supported()` 를 앱이 실행될 때 호출해 주면 됩니다.

`create_base_gui()` 함수 끝부분에서 위 함수를 호출합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
Evas_Object * box, *btn;

/* A box to put things in vertically - default mode for box */
box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,

```

```

EVAS_HINT_EXPAND);
    elm_object_content_set(ad->conform, box);
    evas_object_show(box);

    { /* child object - indent to how relationship */
        /* Label-0 */
        ad->label0 = elm_label_add(ad->conform);
        elm_object_text_set(ad->label0, "Msg - ");
        my_box_pack(box, ad->label0, 1.0, 0.0, -1.0, 0.0);

        /* Label-1 */
        ad->label1 = elm_label_add(ad->conform);
        elm_object_text_set(ad->label1, "Value - ");
        my_box_pack(box, ad->label1, 1.0, 1.0, -1.0, 0.0);

    }
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);
}

```

Box와 2개의 Label 위젯을 생성하였습니다. 그리고 센서 지원여부 판단 함수를 호출했습니다.

예제를 빌드하고 실행시켜 봅시다. Proximity 센서가 지원되면 'Proximity Sensor is support'라는 문구가 표시됩니다. 스마트 폰 중에서 센서가 지원되지 않는 경우도 있습니다. 그런 경우에는 에뮬레이터에서 테스트하기 바랍니다.

Proximity Sensor is support

Value -

2) Proximity 센서 이벤트 구하기

Proximity 센서에 물체가 감지되면 그 이벤트를 구해서 거리 수치를 화면에 표시하는 기능을 구현해 보겠습니다. 소스파일 위쪽에 센서 관련 구조체와 전역변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label0;
    Evas_Object *label1;
} appdata_s;

typedef struct _sensor_info
{
    sensor_h sensor;      /**< Sensor handle */
    sensor_listener_h sensor_listener;
} sensorinfo;

static sensorinfo sensor_info;
```

sensorinfo는 센서 객체와 이벤트 리스너 변수를 포함하고 있는 구조체입니다.

sensor_info는 sensorinfo 구조체의 전역변수입니다.

센서 이벤트를 구하는 것은 리스너를 시작하는 것입니다. 센서 객체와 이벤트 리스너를 사용해서 Proximity 센서 이벤트를 구해보겠습니다. `create_base_gui()` 함수 위쪽에 새로운 함수 2개를 생성합니다.

```
static void  
_new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void *user_data)  
{  
    if( sensor_data->value_count < 1 )  
        return;  
    char buf[PATH_MAX];  
    appdata_s *ad = (appdata_s*)user_data;  
  
    sprintf(buf, "Distance : %0.1f", sensor_data->values[0]);  
    elm_object_text_set(ad->label1, buf);  
}  
  
static void  
start_proximity_sensor(appdata_s *ad)  
{  
    sensor_error_e err = SENSOR_ERROR_NONE;  
    sensor_get_default_sensor(SENSOR_PROXIMITY, &sensor_info.sensor);  
    err = sensor_create_listener(sensor_info.sensor, &sensor_info.sensor_listener);  
    sensor_listener_set_event_cb(sensor_info.sensor_listener, 100, _new_sensor_value, ad);  
    sensor_listener_start(sensor_info.sensor_listener);  
}
```

`_new_sensor_value()` 는 Proximity 센서 이벤트 콜백 함수입니다. 새로운 센서 값을 화면에 출력합니다.

2번째 파라미터에 센서 데이터가 전달됩니다. `values[0]`에 거리 데이터가 저장되어 있습니다.

`start_proximity_sensor()` 는 Proximity 센서를 시작하고, 이벤트 콜백 함수를 지정하는 함수입니다.

`sensor_get_default_sensor(sensor_type_e, sensor_h *)` 는 센서 객체를 반환하는 API입니다. 1번째 파라미터에 SENSOR_PROXIMITY 를 전달하면, 2번째 파라미터에 Proximity 센서 객체가 반환됩니다.

`sensor_create_listener(sensor_h, sensor_listener_h *)` 는 이벤트 리스너를 생성하는 API입니다. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

`sensor_listener_set_event_cb(sensor_listener_h, unsigned int, sensor_event_cb, void *)` 는 리스너에 콜백 함수를 지정하는 API입니다. 파라미터는 순서대로 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

`sensor_listener_start(sensor_listener_h)` 는 리스너를 시작하는 API 입니다.

앱이 실행되면 자동으로 이벤트 리스너를 가동시켜 보겠습니다.
`create_base_gui()` 함수 끝부분에 위 함수를 호출해 줍니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

show_is_supported(ad);
start_proximity_sensor(ad);

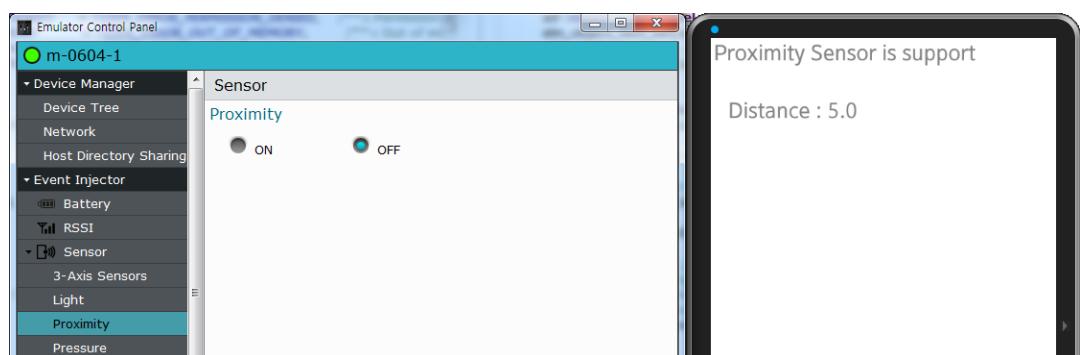
}
```

예제를 다시 실행시켜 보겠습니다. 스마트폰에서 테스트 할 때는 폰을 얼굴에 가까이 가져가면 됩니다. 에뮬레이터에서 테스트 할 때는 Control Panel을 사용하면 됩니다.

에뮬레이터를 마우스 오른쪽 버튼으로 클릭하고 단축 메뉴에서 Control Panel을 선택합니다.

Control Panel이 나타나면 왼쪽 트리 목록에서 [Event Injector > Proximity]를 선택합니다.

Control Panel 오른쪽 화면에서 ON 버튼을 누르면 앱 화면의 2번째 Label 위젯에 0.0 이 표시됩니다. OFF 버튼을 누르면 Label 위젯에 5.0 이 표시됩니다.



3) 관련 API

`int sensor_is_supported(sensor_type_e type, bool *supported)` : 특정 센서의 지원여부를 구하는 API. 1번째 파라미터에 SENSOR_PROXIMITY를 전달하면, 2번째 파라미터에서 Proximity 센서 지원 여부를 반환해 줍니다.

`int sensor_get_default_sensor(sensor_type_e type, sensor_h`

*sensor) : 센서 객체를 반환하는 API. 1번째 파라미터에 SENSOR_PROXIMITY를 전달하면, 2번째 파라미터에 Proximity 센서 객체가 반환됩니다.

int sensor_create_listener(sensor_h sensor, sensor_listener_h *listener) : 이벤트 리스너를 생성하는 API. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

int sensor_listener_set_event_cb(sensor_listener_h listener, unsigned int interval_ms, sensor_event_cb callback, void *data) : 리스너에 콜백 함수를 지정하는 API. / 파라미터 : 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

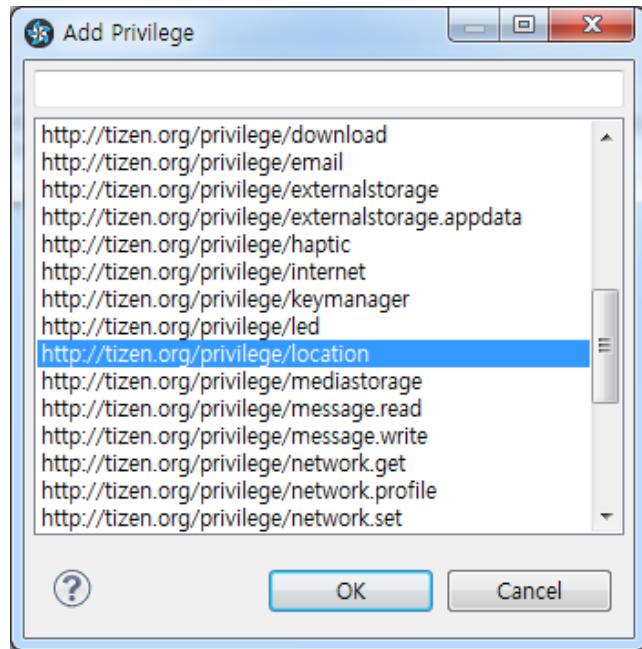
int sensor_listener_start(sensor_listener_h listener) : 리스너를 시작하는 API.

56. GPS 센서 사용방법

지도 앱 또는 네비게이션 앱을 구현하려면 위치좌표를 알아야 합니다. 페이스북, 트위터 같은 SNS와 주변 정보 제공 앱 같은 위치기반 서비스도 위치좌표는 필수 사항입니다. 이번 예제에서는 Location Manager를 통해서 GPS 센서를 사용하는 방법을 알아보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 SensorGps 으로 지정합니다. Location Manager를 사용하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/location> 을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



저장하고 아래쪽 탭버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.sensorgps" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.sensorgps" exec="sensorgps" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>sensorgps</label>
        <icon>sensorgps.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/location</privilege>
    </privileges>
</manifest>
```

2) Location Manager 상태 체크

Location Manager 가 사용 가능한 상태인지를 확인해 보겠습니다. src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
-----  
#include "sensorgps.h"  
#include <locations.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label0;  
    Evas_Object *label1;  
    location_manager_h manager;  
} appdata_s;  
-----
```

locations.h는 Location Manager 라이브러리 헤더파일입니다.

label0에는 Location Manager의 상태를 표시하고, label1에는 위치 정보를 표시하겠습니다.

location_manager_h는 Location Manager 구조체입니다.

create_base_gui() 함수 위에 새로운 함수를 2개 생성합니다.

```
-----  
static void  
state_changed_cb(location_service_state_e state, void *user_data)  
{  
    appdata_s *ad = user_data;  
    char buf[100];  
    char *enable = (state == LOCATIONS_SERVICE_ENABLED) ? "Enable" : "Disable";  
-----
```

```

        sprintf(buf, "State is %s", enable);
        elm_object_text_set(ad->label0, buf);
    }

static void
show_state(appdata_s *ad)
{
    location_manager_create(LOCATIONS_METHOD_GPS, &ad->manager);
    location_manager_set_service_state_changed_cb(ad->manager, state_changed_cb, ad);
    location_manager_start(ad->manager);
}

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
}

```

```
/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);
/* show the frame */
evas_object_show(frame);
}
```



state_changed_cb() 는 Location Manager 상태 변경 이벤트 콜백 함수입니다. 1번째 파라미터에는 상태값이 전달됩니다. 상태 종류는 다음과 같습니다.

- LOCATIONS_SERVICE_DISABLED : 서비스 불가능 상태
- LOCATIONS_SERVICE_ENABLED : 서비스 가능 상태

show_state() 는 Location Manager 상태 변경 이벤트를 요청하는 함수입니다.

location_manager_create(location_method_e, location_manager_h*) 는 Location Manager 객체를 생성하는 API입니다. 1번째 파라미터에 LOCATIONS_METHOD_GPS를 전달하면, 2번째 파라미터에서 Location Manager 객체를 반환해 줍니다. 위치 정보 수집 종류는 다음과 같습니다.

- LOCATIONS_METHOD_GPS : GPS 사용
- LOCATIONS_METHOD_WPS : WiFi 사용
- LOCATIONS_METHOD_HYBRID : GPS와 WiFi 중에서 자동 선택

location_manager_set_service_state_changed_cb(location_manager_h, location_service_state_changed_cb, void *) 는 Location Manager 상태 변경 이벤트 함수명을 지정하는 API입니다. 파라미터는 순서대로 Location Manager 객체, 이벤트 콜백 함수명, 사용자 데이터입니다.

location_manager_start(location_manager_h) 는 Location Manager를

기동하는 API입니다.

my_box_pack() 은 Box에 위젯을 추가하는 함수입니다.

앱이 실행되면 자동으로 Location Manager를 기동시켜 보겠습니다.
create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
Evas_Object * box, *btn;

/* A box to put things in vertically - default mode for box */
box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
/* Label-0 */
ad->label0 = elm_label_add(ad->conform);
elm_object_text_set(ad->label0, "Hello EFL");
my_box_pack(box, ad->label0, 1.0, 0.0, -1.0, 0.0);
}
}
```

```

/* Show window after base gui is set up */
evas_object_show(ad->win);

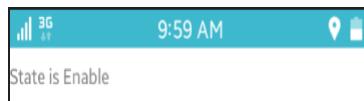
show_state(ad);

}

```

Box와 Label 위젯을 생성하고, Location Manager 상태 변경 이벤트 요청 함수를 호출합니다.

예제를 빌드하고 실행시켜 보겠습니다. Label 위젯에 'State is Enable'라고 표시되면 GPS가 정상동작 하는 것입니다.



3) 현재 위치좌표 구하기

Button을 누르면 현재 위경도 좌표를 구해서 화면에 표시하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```

{ /* child object - indent to how relationship */
  /* Label-0 */
  ad->label0 = elm_label_add(ad->conform);
  elm_object_text_set(ad->label0, "Hello EFL");
  my_box_pack(box, ad->label0, 1.0, 0.0, -1.0, 0.0);

  /* Label-1 */
  ad->label1 = elm_label_add(ad->conform);
  elm_object_text_set(ad->label1, "Hello EFL");

```

```

my_box_pack(box, ad->label1, 1.0, 0.0, -1.0, 0.0);

/* Button */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Get Location");
evas_object_smart_callback_add(btn, "clicked", btn_clicked_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, 0.0);
}

}

```

Label 위젯과 Button 위젯을 1개씩 추가하였습니다.

Button 콜백 함수를 생성하겠습니다. create_base_gui() 함수 위에 새로운 코드를 추가합니다.

```

static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    double altitude, latitude, longitude, climb, direction, speed;
    double horizontal, vertical; location_accuracy_level_e level; time_t timestamp;

    location_manager_get_location(ad->manager, &altitude, &latitude,
                                  &climb, &direction, &speed, &level, &horizontal,
                                  &vertical, &timestamp);
    char buf[100];
    sprintf(buf, "%0.5f/%0.5f", latitude, longitude);
    elm_object_text_set(ad->label1, buf);
}

```

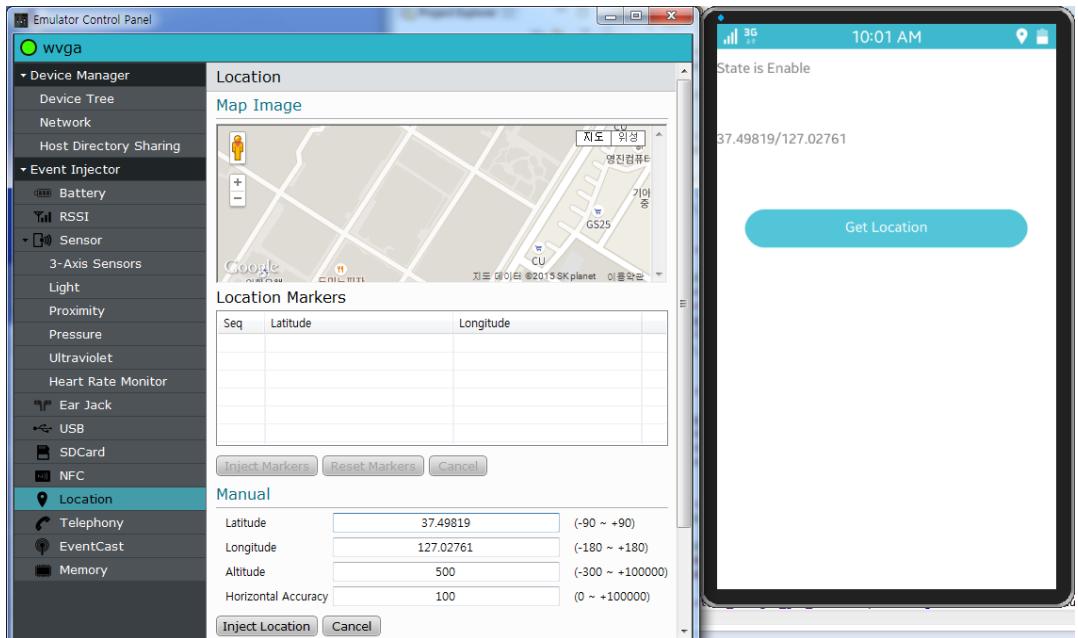
location_manager_get_location(location_manager_h, double *, double *,

`double *, double *, double *, double *, location_accuracy_level_e *,
double *, double*, time_t*)` 는 현재 위치정보를 구하는 API입니다.
파라미터는 순서대로 Location Manager 객체, 고도, 위도, 경도, 수직
이동 속도, 방향, 수평 이동 속도, 정확도, 수평 정확도(단위는 Meters),
수직 정확도(단위는 Meters), 시각입니다.

그 다음은 위도와 경도 좌표를 2번째 Label 위젯에 출력하는
코드입니다.

예제를 다시 실행시켜 봅시다. 에뮬레이터에서 테스트 하려면 Control
Panel을 사용하면 됩니다. 에뮬레이터를 마우스 오른쪽 버튼으로
클릭하고 단축 메뉴에서 Control Panel을 선택합니다.

Control Panel이 나타나면 왼쪽 트리 목록에서 [Event Injector >
Location]을 선택합니다. 그런 다음 오른쪽 화면에서 Latitude에 위도
좌표를 입력하고(ex 37.49819) Longitude에 경도 좌표를 입력합니다.(ex
127.02761) Altitude에는 500, Horizontal Accuracy에는 100 정도
입력합니다. 그런 다음 Inject Location 버튼을 누릅니다. 앱 화면에서
Button을 누르면 Control Panel에서 입력한 위경도 좌표가 2번째 Label
위젯에 표시됩니다.



4) 위치 이동 이벤트 구하기

사용자의 위치가 변경되면 자동으로 새로운 위경도 좌표를 화면에 표시해 보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드 한줄을 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

show_state(ad);
location_manager_set_position_updated_cb(ad->manager, position_updated_cb, 2,
ad);
}
```

`location_manager_set_position_updated_cb(location_manager_h,`

`location_position_updated_cb`, `int`, `void *`) 는 위치 정보 변경 이벤트 요청 API입니다. 파라미터는 순서대로 Location Manager 객체, 이벤트 콜백 함수명, 시간 간격, 사용자 데이터 입니다.

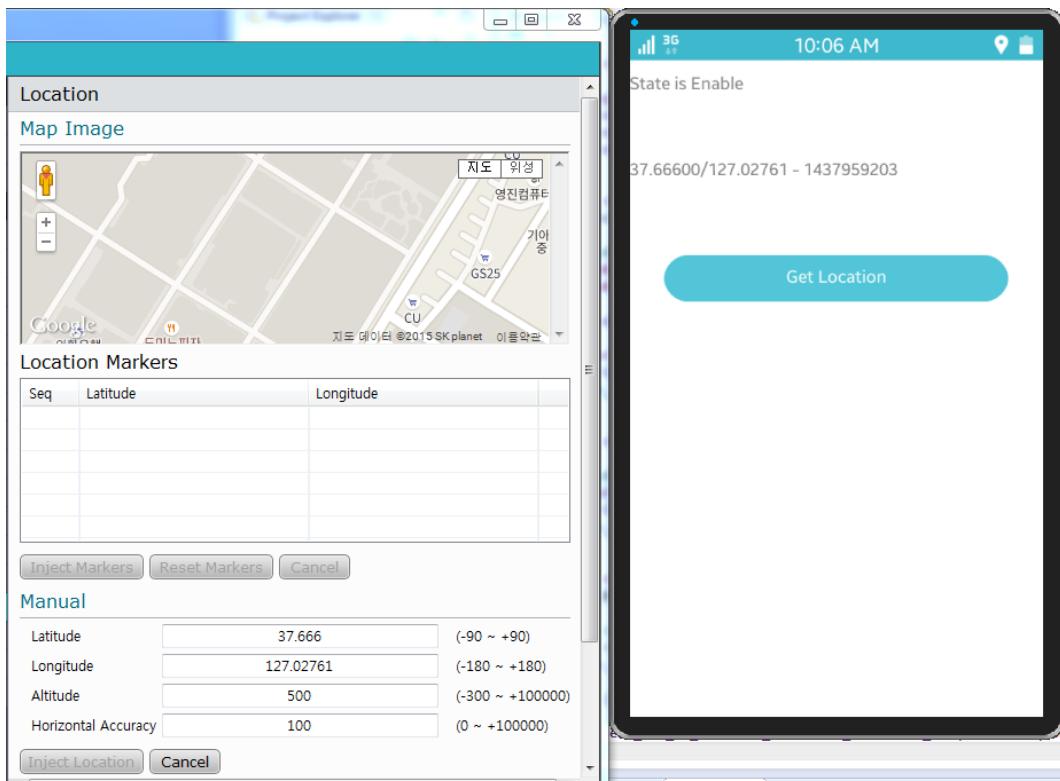
마지막으로 콜백 함수를 생성할 차례입니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void position_updated_cb(double latitude, double longitude, double altitude, time_t timestamp, void *user_data)
{
    appdata_s *ad = user_data;
    char buf[100];
    sprintf(buf, "%0.5f/%0.5f - %ld", latitude, longitude, timestamp);
    elm_object_text_set(ad->label1, buf);
}
```

`position_updated_cb()` 는 Location Manager에 새로운 위치 정보가 수신되었을 때 호출되는 콜백 함수입니다. 파라미터는 순서대로 위도, 경도, 고도, 시각, 사용자 데이터 입니다.

함수 내부에서는 위도와 경도좌표, 그리고 시각을 화면에 출력합니다.

예제를 다시 실행합니다. Control Panel에서 Latitude(ex 37.666) 와 Longitude(ex 127.02761) 값을 변경하고 Inject Location 버튼을 눌러봅시다. 앱 화면에 위경도 좌표가 자동으로 변경됩니다.



5) 관련 API

`int location_manager_create(location_method_e method,
location_manager_h* manager)` : Location Manager 객체를 생성하는 API.
1번째 파라미터에 LOCATIONS_METHOD_GPS를 전달하면, 2번째
파라미터에서 Location Manager 객체를 반환해 줍니다. 위치 정보 수집
종류는 다음과 같습니다.

- LOCATIONS_METHOD_GPS : GPS 사용
- LOCATIONS_METHOD_WPS : WiFi 사용
- LOCATIONS_METHOD_HYBRID : GPS와 WiFi 중에서 자동 선택

`int location_manager_set_service_state_changed_cb(location_manager_h
manager, location_service_state_changed_cb callback, void`

*user_data) : Location Manager 상태 변경 이벤트 함수명을 지정하는 API. 파라미터는 순서대로 Location Manager 객체, 이벤트 콜백 함수명, 사용자 데이터입니다.

int location_manager_start(location_manager_h manager) : Location Manager를 기동하는 API.

int location_manager_get_location(location_manager_h manager, double *altitude, double *latitude, double *longitude, double *climb, double *direction, double *speed, location_accuracy_level_e *level, double *horizontal, double *vertical, time_t *timestamp) : 현재 위치정보를 구하는 API. 파라미터는 순서대로 Location Manager 객체, 고도, 위도, 경도, 수직 이동 속도, 방향, 수평 이동 속도, 정확도, 수평 정확도(단위는 Meters), 수직 정확도(단위는 Meters), 시각입니다.

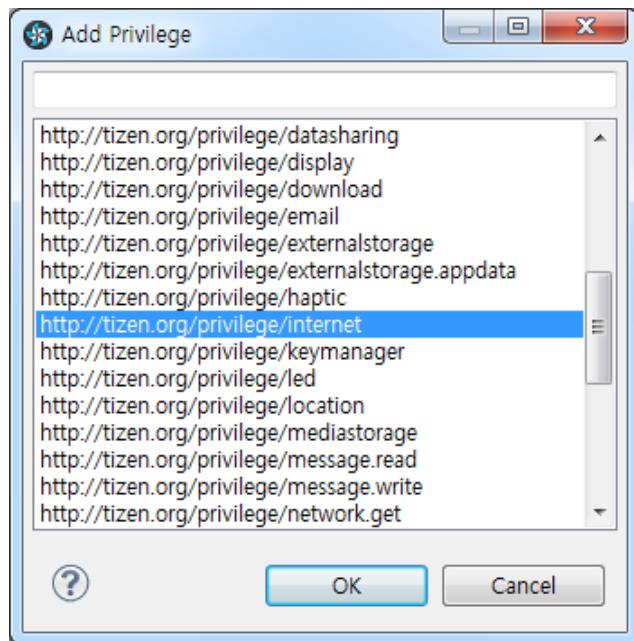
int location_manager_set_position_updated_cb(location_manager_h manager, location_position_updated_cb callback, int interval, void *user_data) : 위치 정보 변경 이벤트 요청 API. 파라미터는 순서대로 Location Manager 객체, 이벤트 콜백 함수명, 시간 간격, 사용자 데이터입니다.

57. 구글맵 라이브러리

전자지도를 사용하려면 구글 맵을 사용하면 됩니다. 방법은 구글 맵 서버에 위경도 좌표와 Zoom 레벨값을 전달하고, 서버에서 전달되는 지도 데이터를 캔버스에 입력하면 됩니다. 예제를 하드코딩으로 작성하려면 시간이 오래 걸리기 때문에 라이브러리 파일을 불러와서 간편하게 구현해 보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 MapViewEx 으로 지정합니다. 구글 맵 서버와 통신하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/internet> 을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



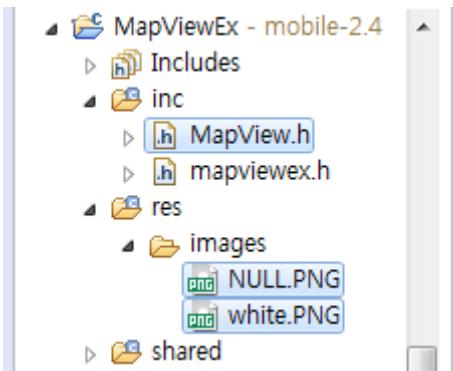
저장하고 아래쪽 텁버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.mapviewex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.mapviewex" exec="mapviewex" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>mapviewex</label>
        <icon>mapviewex.png</icon>
    </ui-application>
    <privileges>
        <b><privilege>http://tizen.org/privilege/internet</privilege></b>
    </privileges>
</manifest>
```

2) 라이브러리 파일 복사

전자지도 기능을 모두 하드코딩으로 구현하려면 너무 길어지기 때문에 라이브러리 파일을 복사해서 사용하도록 하겠습니다. 부록 /etc 폴더에 있는 MapView.h 파일을 소스 프로젝트 /inc 폴더로 복사합니다.

지도 소스코드에서는 이미지 파일도 사용합니다. 소스 프로젝트 /res 폴더에 새로운 폴더를 만들고 이름을 images으로 지정합니다. 그런 다음, 부록 /Image 폴더에서 2개의 이미지 파일(NULL.PNG, white.PNG) 방금 생성한 폴더로 복사합니다.



3) MapView 위젯 생성

라이브러리와 필요한 이미지 복사가 완료되었으니 소스코드로 MapView 위젯을 생성하겠습니다. /src 폴더에 있는 소스파일(~.c)을 열고 라이브러리와 위경도 좌표를 추가합니다.

```
#include "mapviewex.h"
#include "MapView.h"

#define START_LATITUDE 40.779986
#define START_LONGITUDE -73.9615488
```

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
} appdata_s;
```

MapView.h 는 좀전에 부록에서 복사한 MapView 라이브러리 파일입니다.

START_LATITUDE와 START_LONGITUDE는 뉴욕 맨하튼 센트럴 파크의 위경도 좌표입니다. 본인이 사용하고 싶은 좌표로 대신해도 상관없습니다.

create_base_gui() 함수에 새로운 코드를 추가합니다. Label 위젯은 주석 처리합니다.

```
/* Label*/  
/*ad->label = elm_label_add(ad->conform);  
elm_object_text_set(ad->label, "Hello EFL");  
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
elm_object_content_set(ad->conform, ad->label);  
evas_object_show(ad->label);*/  
  
create_map(ad->conform, START_LATITUDE, START_LONGITUDE);  
  
/* Show window after base gui is set up */  
evas_object_show(ad->win);
```

create_map(Evas_Object *, double, double) 는 MapView 위젯을 생성하는 함수입니다. 파라미터는 순서대로 컨테이너, 위도 좌표, 경도 좌표

입니다.

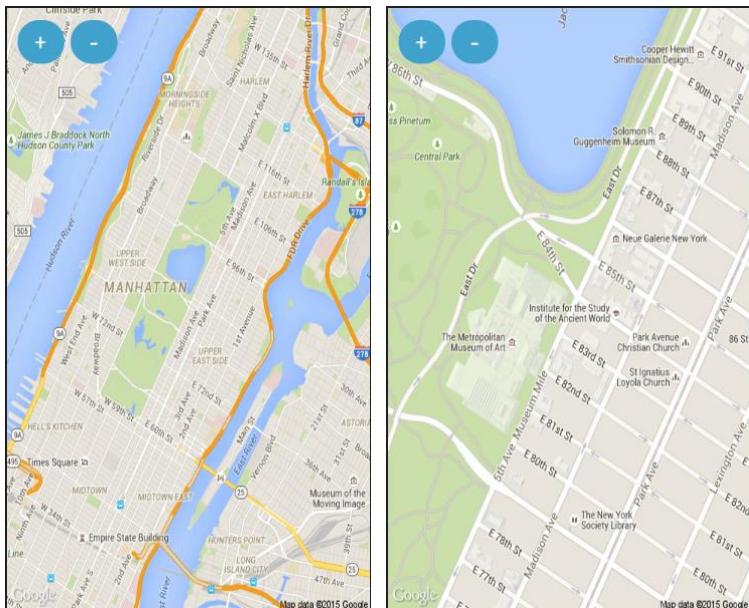
예제를 빌드하고 실행시켜 봅시다. 만약 빌드하는 도중에 오류가 발생한다면 /inc/MapView.h 파일을 열고

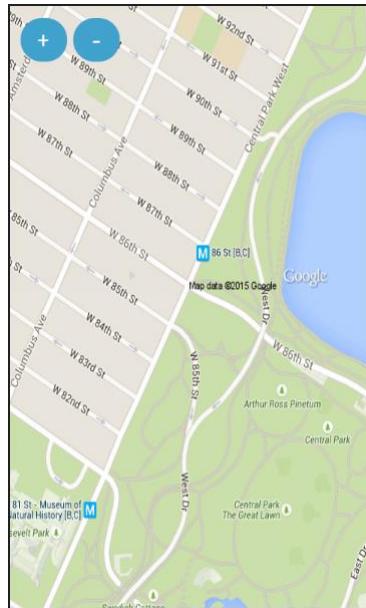
```
#include <curl/curl.h>
```

위 코드를 아래와 같이 수정하기 바랍니다.

```
#include <curl.h>
```

전자지도가 표시되고 2개의 버튼이 보입니다. + 버튼을 누르면 지도가 확대되고, - 버튼을 누르면 축소됩니다. 지도를 드래그하면 그 방향으로 지도가 이동합니다.





4) MapView 위젯 소스코드 설명

MapView 라이브러리에서 사용하는 소스코드에 대해서 알아보겠습니다.
/inc 폴더에 복사했던 MapView.h 파일을 열고 가장 아래쪽으로
이동하면 create_map()의 끝에 다음과 같은 코드가 있습니다.

int mkdir (char * __path, __mode_t __mode) : 새로운 폴더를 생성하는 API.

make_new_url() 함수에서는 위경도 좌표로 구글 맵 서버에 지도
데이터를 요청하는 URL 경로를 생성합니다. 생성된 URL 주소는
전역변수 place_api_url 혹은 curr_url에 저장됩니다.

arrange_start_main_page() 함수에서는 구글 맵 서버에 위경도 좌표를
전달하는 역할을 합니다.

map_dload_thread() 는 지도 데이터 다운로드 스레드 이벤트

함수입니다.

save_map_temp_file() 는 서버에서 전송받은 지도 데이터를 파일로 저장하는 함수입니다.

update_main_page() 는 지도 데이터를 타일에 저장하고 타일의 위치와 크기를 변경하는 함수입니다.

mouse_down_cb() 는 사용자가 지도 타일을 Touch 다운했을 때 이벤트를 처리하는 함수입니다.

mouse_move_cb() 는 사용자가 지도 타일을 Touch 이동했을 때 지도 타일을 이동하는 함수입니다.

mouse_up_cb() 는 사용자가 지도 타일을 Touch 해제했을 때 이벤트를 처리하는 함수입니다.

map_zoom_in() 는 + 버튼의 콜백 함수입니다. Zoom 레벨을 증가하고 중심 좌표를 변경해서 arrange_start_main_page() 함수를 호출합니다.

map_zoom_out() 는 - 버튼의 콜백 함수입니다. Zoom 레벨을 감소하고 중심 좌표를 변경해서 arrange_start_main_page() 함수를 호출합니다.

5) MapView.h 소스코드

부록 파일을 다운받지 못한 경우에는 텍스트 파일을 생성해서 파일명을 MapView.h로 지정합니다. 그런 다음 아래 코드를 복사하면 됩니다.

define 구문 중에서 START_URL 과 PLACE_API_URL 끝부분에 있는 <Google Map Key> 는 자신이 소유하고 있는 구글 맵 Key를 입력하면

됩니다.

```
/*
 *MapView.h
 *
 * Created on: Apr 29, 2015
 *      Author: 김시찬 Samsung Electronics co
 */

#ifndef MAPVIEW_H_
#define MAPVIEW_H_

#include <app.h>
#include <Evas_GL.h>
#include <Evas_GL_GLES2_Helpers.h>
#include <Elementary.h>
#include <system_settings.h>
#include <efl_extension.h>
#include <dlog.h>
#include <curl/curl.h>
#include <math.h>
#include <dlog.h>
#include <locations.h>

#define TMP_DIR "/tmp/map_temp"
#define START_URL
"http://maps.googleapis.com/maps/api/staticmap?language=korean&zoom=99&center
=unknown&size=480x800&key=<Google Map Key>"
#define PLACE_API_URL
"https://maps.googleapis.com/maps/api/place/textsearch/xml?query=where&sensor=true&key=<Google Map Key>"

#define MAX_TMP_FILENAME_LEN 256
#define MAX_URL_LEN 2048
```

```
#define MAP_TILE_X 3
#define MAP_TILE_Y 3
#define MAX_ZOOM_SCALE 19
#define MIN_ZOOM_SCALE 0
#define START_ZOOM_LEVEL 13
#define X_RESOLUTION 480
#define Y_RESOLUTION 800
#define ALL_TILE_USABLE_X 1000
#define ALL_TILE_USABLE_Y 1000
#define GEO_INFO_STR_NUM 20
#define MAP_DLOAD_Q_NUM 500
#define PLACE_SEARCHED_NUM 100
#define PLACE_INFO_MAX (100*1000)
```

```
typedef struct mapdata {
    float          xangle;
    float          yangle;
    Eina_Bool      mouse_down : 1;
    Eina_Bool      mouse_move : 1;
    Eina_Bool      mouse_move_update : 1;
    Eina_Bool      wait_for_update : 1;
    Ecore_Thread*  thread;
} mapdata_s;
```

```
typedef struct all_tile_info {
    double lati_of_center;
    double longti_of_center;
    char download_ok;
    char download_request;
    char file_name[MAX_TMP_FILENAME_LEN];
}all_tile_info_s;
```

```
typedef struct index_of_all_tile {
    int x;
    int y;
```

```

}index_of_all_tile_s;

typedef struct certer_tile_info {
    int x_info;
    int y_info;
}certer_tile_info_s;

typedef struct map_dload_queue {
    char url[MAX_URL_LEN];
    char filename[MAX_TMP_FILENAME_LEN];
    index_of_all_tile_s index;
}map_dload_queue_s;

typedef struct place_search_list {
    double str_lati;
    double str_longti;
} place_search_list_s;

typedef struct tile_info {
    int curr_x;
    int curr_y;
    char file_name[MAX_TMP_FILENAME_LEN];
}tile_info_s;

typedef struct MemoryStruct {
    char *memory;
    size_t size;
} MemoryStruct_s;

enum
{
    URL_CHANGED,
    URL_NOT_CHANGED,
}new_url_result;

```

```

enum
{
    ZOOM_CHANGED,
    CENTER_NAME_CHANGED,
    CENTER_GEOMETRY_CHANGED,
    ADD_NEW_MARKER,
    ADD_NEW_MULTI_MARKER,
    DELITE_MARKER,
    PLACE_INFO,
    MAX_REASON
}new_url_reason;

const tile_info_s tile_info_init_information[MAP_TILE_X][MAP_TILE_Y] = {
    {{-X_RESOLUTION,Y_RESOLUTION},      {-X_RESOLUTION,0},      {-X_RESOLUTION,-
Y_RESOLUTION}},
    {{0,Y_RESOLUTION}, {0,0}, {0,-Y_RESOLUTION}},
    {{X_RESOLUTION,Y_RESOLUTION},       {X_RESOLUTION,0},      {X_RESOLUTION,-
Y_RESOLUTION}}
};

tile_info_s tile_info[MAP_TILE_X][MAP_TILE_Y];

//appdata_s *ad_g;
mapdata_s *m_md;
Evas *m_canvas;
Evas_Object* main_page[MAP_TILE_X][MAP_TILE_Y];
Evas_Object *m_btn1;
Evas_Object *m_btn2;

int map_dload_q_idx = 0;
char do_not_update_map = 0;
map_dload_queue_s map_dload_q[MAP_DLOAD_Q_NUM];
all_tile_info_s all_tiles[ALL_TILE_USABLE_X][ALL_TILE_USABLE_Y];
char place_api_url[MAX_URL_LEN] = PLACE_API_URL;
char curr_url[MAX_URL_LEN] = START_URL;
char search_str[MAX_REASON][20]={"zoom=", "center=", "center=", "&", "&",

```

```

"&markers=", "query="};

int place_searched_num;
place_search_list_s place_search_list[PLACE_SEARCHED_NUM];
location_manager_h l_manager;
int location_initialized = 0;
double map_start_lati_of_center = 37.259606;
double map_start_longti_of_center = 127.045828;
double x_moved = 0;
double y_moved = 0;
Ecore_Timer * get_geometry_timer = NULL;
static char temp_place_info[PLACE_INFO_MAX];

Eina_Lock      set_info_mutex;
extern int map_dload_q_idx;
int current_zoom_level = START_ZOOM_LEVEL;
certer_tile_info_s current_center;
double curr_user_lati;
double curr_user_longti;
char m_icon_path[100];

void update_main_page(void);
Eina_Bool get_changed_location(void *data);

double next_lati_value(double curr_lati, int diff, int curr_zoom)
{
    double lat;

    int y1 = floor((1.0 - log(tan(curr_lati * ELM_PI / 180.0) +
        (1.0 / cos(curr_lati * ELM_PI / 180.0))) /
        ELM_PI) / 2.0 * (Y_RESOLUTION*pow(2, curr_zoom)));

    double n = ELM_PI - (2.0 * ELM_PI * (y1+2.5*diff) / (Y_RESOLUTION*pow(2,
curr_zoom)));
    lat = 180.0 / ELM_PI *atan(0.5 * (exp(n) - exp(-n)));
}

```

```

    return lat;
}

double next_longti_value(double curr_longti, int diff, int curr_zoom)
{
    double lon;

    int x1 = floor((curr_longti + 180.0) / 360.0 * (X_RESOLUTION*pow(2, curr_zoom)));

    lon = ((x1+1.875*diff) / ((double)X_RESOLUTION*pow(2, curr_zoom)) * 360.0 - 180;

    return lon;
}

double latitude_of_polar(int polar)// polar=1 : Arctic, polar=-1 : Antarctic
{
    double n, y, lat;

    if(polar==1)
        y = 0;
    else
        y = Y_RESOLUTION;
    n = ELM_PI - (2.0 * ELM_PI * y / Y_RESOLUTION);
    lat = 180.0 / ELM_PI *atan(0.5 * (exp(n) - exp(-n)));
    return lat;
}

void delete_all_marker_from_url(char* url)
{
    char tmp_url_1[MAX_URL_LEN]={0}, tmp_url_2[MAX_URL_LEN]={0,};
    char* next_pos = NULL;
    char* pos_and_oper = NULL;

    next_pos = strstr(url, "&markers=");
}

```

```

while(next_pos!=NULL)
{
    pos_and_oper = strstr((next_pos+1), "&");
    memset(tmp_url_1, 0, MAX_URL_LEN);
    memcpy(tmp_url_1, url, (next_pos-url));
    memset(tmp_url_2, 0, MAX_URL_LEN);
    memcpy(tmp_url_2, pos_and_oper, strlen(url)-(pos_and_oper-url));
    memset(url, 0, MAX_URL_LEN);
    sprintf(url, MAX_URL_LEN, "%s%s", tmp_url_1, tmp_url_2);
    next_pos = strstr(url, "&markers=");
}

int make_new_url(int reason, int zoom, void* data, double x_changed, double y_changed)
{
    char prev[MAX_URL_LEN]={0}, new_url[MAX_URL_LEN]={0},
        tmp_str_changed[MAX_URL_LEN]={0}, tmp_str_changed_1[MAX_URL_LEN]={0},
        tmp_str_end[MAX_URL_LEN]={0};
    char *url_handle;
    char str_lati[GEO_INFO_STR_NUM]={0}, str_longi[GEO_INFO_STR_NUM]={0};
    char* pos = NULL;
    char* pos_and_oper = NULL;
    char* pos_comma = NULL;
    double latitude, longitude;

    if(reason == PLACE_INFO)
    {
        url_handle = place_api_url;
    }
    else
    {
        url_handle = curr_url;
    }
    memcpy(prev, url_handle, strlen(url_handle));
}

```

```

pos = strstr(prev, search_str[reason]);

if(pos == NULL)
{
    return URL_NOT_CHANGED;
}

pos_and_oper = strstr(pos, "&"); // to search first '&' from after "zoom=..."

if(pos_and_oper == NULL)
{
    return URL_NOT_CHANGED;
}

memcpy(tmp_str_end, pos_and_oper, strlen(prev) - ((int)pos_and_oper-(int)prev));

switch (reason)
{
    case ZOOM_CHANGED:
    {
        if(zoom>0)
        {
            if(current_zoom_level<MAX_ZOOM_SCALE) current_zoom_level++;
            else return URL_NOT_CHANGED;
        }
        else if(zoom<0)
        {
            if(current_zoom_level>MIN_ZOOM_SCALE) current_zoom_level--;
            else return URL_NOT_CHANGED;
        }
        snprintf(tmp_str_changed,      sizeof(tmp_str_changed),      "zoom=%d",
current_zoom_level);
    }
    break;
}

```

```

case CENTER_NAME_CHANGED:
{
    snprintf(tmp_str_changed, sizeof(tmp_str_changed), "center=%s", (char*)data);
}
break;

case CENTER_GEOMETRY_CHANGED:
{
    pos_comma = strstr(pos, ","); // to search first ',' from after "center=..."
    memcpy(str_lati, (char*)((int)pos+strlen("center=")), (int)pos_comma - ((int)pos+strlen("center=")));
    memcpy(str_longi, (char*)((int)pos_comma+1), ((int)pos_and_oper-((int)pos_comma+1)) );
    latitude = atof(str_lati);
    longitude = atof(str_longi);

    if(latitude>90) latitude=90;
    if(latitude<-90) latitude=-90;

    if(longitude>180) longitude=(-1)*(360-longitude);
    if(longitude<-180) longitude=(longitude+360);

    snprintf(tmp_str_changed, sizeof(tmp_str_changed), "center=%f,%f", latitude, longitude);
}
break;

case ADD_NEW_MARKER:
{
    char tmp_char = '%';
    snprintf(tmp_str_changed, sizeof(tmp_str_changed),
    "&markers=color:blue%7Clabel:U%7C%f,%f", tmp_char, tmp_char, y_changed,
    x_changed);
}

```

```

break;

case ADD_NEW_MULTI_MARKER:
{
    char tmp_char = '%';
    int i = place_searched_num;

    while(i--)
    {
        snprintf(tmp_str_changed_1, sizeof(tmp_str_changed_1),
"%s&markers=color:red%c7Clabel:S%c7C%f,%f", tmp_str_changed, tmp_char, tmp_char,
place_search_list[i].str_lati, place_search_list[i].str_longti);
        snprintf(tmp_str_changed, sizeof(tmp_str_changed), "%s",
tmp_str_changed_1);
    }
}
break;

case DELITE_MARKER:
{
    //do not process tmp_str_changed because we just need to set
tmp_str_changed=""
    delete_all_marker_from_url(tmp_str_end);// remove all "&markers.." from
tmp_str_end
}
break;

case PLACE_INFO:
{
    snprintf(tmp_str_changed, sizeof(tmp_str_changed), "query=%s", (char*)data);
}
break;

default :
break;

```

```

}

memcpy(&new_url[0], prev, ((int)pos-(int)prev));
memcpy(&new_url[((int)pos-(int)prev)], tmp_str_changed, strlen(tmp_str_changed));
memcpy(&new_url[((int)pos-(int)prev)+strlen(tmp_str_changed)],           tmp_str_end,
strlen(tmp_str_end));

memset(url_handle, 0, MAX_URL_LEN);
memcpy(url_handle, new_url, strlen(new_url));

return URL_CHANGED;
}

void request_map_download(int x_info, int y_info, double latitude, double longitude)
{
    char buf[100];

    if(all_tiles[x_info][y_info].download_request)
        return;

    all_tiles[x_info][y_info].download_request = 1;

    memset(all_tiles[x_info][y_info].file_name, 0, MAX_TMP_FILENAME_LEN);

    if(longitude>200     ||     longitude<-200     ||     latitude<latitude_of_polar(-1)     ||
latitude>latitude_of_polar(1))
    {
        snprintf(all_tiles[x_info][y_info].file_name,     sizeof(all_tiles[x_info][y_info].file_name),
"%s/white.PNG", m_icon_path);
        return;
    }

    all_tiles[x_info][y_info].lati_of_center = latitude;
    all_tiles[x_info][y_info].longti_of_center = longitude;
    snprintf(all_tiles[x_info][y_info].file_name,         sizeof(all_tiles[x_info][y_info].file_name),

```

```

"%s/%d_%f.%f.PNG", TMP_DIR, current_zoom_level, latitude, longitude);

memset(buf, 0, 100);
snprintf(buf, sizeof(buf), "%f,%f", latitude, longitude);

make_new_url(CENTER_NAME_CHANGED, 0, buf, 0, 0);

eina_lock_take(&set_info_mutex);
memset(&map_dload_q[map_dload_q_idx].url[0], 0, MAX_URL_LEN);
memcpy(&map_dload_q[map_dload_q_idx].url[0], curr_url, MAX_URL_LEN);
memset(&map_dload_q[map_dload_q_idx].filename[0], 0, MAX_TMP_FILENAME_LEN);
memcpy(&map_dload_q[map_dload_q_idx].filename[0],
all_tiles[x_info][y_info].file_name, MAX_TMP_FILENAME_LEN);
map_dload_q[map_dload_q_idx].index.x = x_info;
map_dload_q[map_dload_q_idx].index.y = y_info;
map_dload_q_idx++;
eina_lock_release(&set_info_mutex);
}

int save_map_temp_file(char* url, char* file_name)
{
    CURL *curl_handle;
    FILE *currfile;
    int ret_val=0;

    curl_global_init(CURL_GLOBAL_ALL);

    /* init the curl session */
    curl_handle = curl_easy_init();

    /* set URL to get */
    curl_easy_setopt(curl_handle, CURLOPT_URL, url);

    /* no progress meter please */
    curl_easy_setopt(curl_handle, CURLOPT_FOLLOWLOCATION, 1L);

```

```

/* open the files */
currfile = fopen(file_name,"w");
if (currfile == NULL) {
    curl_easy_cleanup(curl_handle);
    return -1;
}

/* we want the headers to this file handle */
curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, currfile);

/*
 *      * Notice here that if you want the actual data sent anywhere else but
 *      *      * stdout, you should consider using the CURLOPT_WRITEDATA option.  */
/* get it! */
ret_val = curl_easy_perform(curl_handle);

/* close the header file */
fclose(currfile);

/* cleanup curl stuff */
curl_easy_cleanup(curl_handle);

return ret_val;
}

int map_download(char* tmp_url, char* tmp_filename)
{
    if(save_map_temp_file(tmp_url, tmp_filename) != 0)
    {
        return 0;//fail
    }

    return 1;//success
}

```

```

}

void set_map_main_tile_info(int current_center_x, int current_center_y)
{
    for(int i=0; i<MAP_TILE_X ; i++)
    {
        for(int j=0; j<MAP_TILE_Y ; j++)
        {
            memset(tile_info[i][j].file_name, 0, MAX_TMP_FILENAME_LEN);

            if(all_tiles[current_center_x-(1-i)][current_center_y-(1-j)].download_ok)
            {
                memcpy(tile_info[i][j].file_name, all_tiles[current_center_x-(1-i)][current_center_y-(1-j)].file_name,
                       strlen(all_tiles[current_center_x-(1-i)][current_center_y-(1-j)].file_name));
            }
            else
            {
                snprintf(tile_info[i][j].file_name, sizeof(tile_info[i][j].file_name),
                         "%s/NULL.PNG", m_icon_path);
            }
        }
    }

    current_center.x_info = current_center_x;
    current_center.y_info = current_center_y;
}

void mouse_move_cb(void *data, Evas *e , Evas_Object *obj , void *event_info)
{
    Evas_Event_Mouse_Move *ev;
    ev = (Evas_Event_Mouse_Move *)event_info;
    //appdata_s *ad = data;
    mapdata_s *md = data;

    if(md->mouse_down == EINA_TRUE)

```

```

{
    if((ev->cur.canvas.x != ev->prev.canvas.x) || (ev->cur.canvas.y != ev->prev.canvas.y))
    {
        ecore_timer_del(get_geometry_timer);
    }

    // x point check
    if(ev->cur.canvas.x > ev->prev.canvas.x) //x++
    {
        if(tile_info[1][1].curr_x > 0)
        {
            for(int j=0 ; j<MAP_TILE_Y ; j++)
            {
                if(!
                    all_tiles[current_center.x_info-2][current_center.y_info+(1-j)].download_request)
                {
                    double lati_of_center =
                    all_tiles[current_center.x_info][current_center.y_info].lati_of_center;
                    double longti_of_center =
                    all_tiles[current_center.x_info][current_center.y_info].longti_of_center;
                    double longti_for_end_check;

                    lati_of_center = next_lati_value(lati_of_center, Y_RESOLUTION*(j-1),
                    current_zoom_level);
                    longti_of_center = next_longti_value(longti_of_center,
                    X_RESOLUTION*(-1), current_zoom_level);
                    longti_of_center = next_longti_value(longti_of_center,
                    X_RESOLUTION*(-1), current_zoom_level);

                    request_map_download(current_center.x_info-2,
                    current_center.y_info+(1-j), lati_of_center, longti_of_center);
                }
            }
        }
    }
}

```

```

for(int i=0; i<MAP_TILE_X ; i++)
{
    for(int j=0; j<MAP_TILE_Y ; j++)
    {
        tile_info[i][j].curr_x = tile_info[i][j].curr_x+(ev->cur.canvas.x - ev->prev.canvas.x);
        evas_object_move(main_page[i][j], tile_info[i][j].curr_x, tile_info[i][j].curr_y);
    }
}
x_moved = x_moved + (ev->cur.canvas.x - ev->prev.canvas.x);

if(tile_info[1][1].curr_x > X_RESOLUTION)
{
    set_map_main_tile_info(current_center.x_info-1, current_center.y_info);
    update_main_page();

    for(int i=0; i<MAP_TILE_X ; i++)
    {
        for(int j=0; j<MAP_TILE_Y ; j++)
        {
            evas_object_move(main_page[i][j], tile_info[i][j].curr_x-X_RESOLUTION,
tile_info[i][j].curr_y);
            tile_info[i][j].curr_x = tile_info[i][j].curr_x-X_RESOLUTION;
        }
    }
}
else if(ev->cur.canvas.x < ev->prev.canvas.x) //x--
{
    if(tile_info[1][1].curr_x < 0)
    {
        for(int j=0 ; j<MAP_TILE_Y ; j++)
        {
            if(!all_tiles[current_center.x_info+2][current_center.y_info+(1-j)].download_request)

```

```

{
    double lati_of_center = all_tiles[current_center.x_info][current_center.y_info].lati_of_center;
    double longti_of_center = all_tiles[current_center.x_info][current_center.y_info].longti_of_center;
    double longti_for_end_check;

    lati_of_center = next_lati_value(lati_of_center, Y_RESOLUTION*(j-1),
current_zoom_level);
    longti_of_center = next_longti_value(longti_of_center, X_RESOLUTION,
current_zoom_level);
    longti_of_center = next_longti_value(longti_of_center, X_RESOLUTION,
current_zoom_level);

    request_map_download(current_center.x_info+2,
current_center.y_info+(1-j), lati_of_center, longti_of_center);
}

}

for(int i=0; i<MAP_TILE_X ; i++)
{
    for(int j=0; j<MAP_TILE_Y ; j++)
    {
        tile_info[i][j].curr_x = tile_info[i][j].curr_x+(ev->cur.canvas.x - ev->prev.canvas.x);
        evas_object_move(main_page[i][j], tile_info[i][j].curr_x, tile_info[i][j].curr_y);
    }
}
x_moved = x_moved + (ev->cur.canvas.x - ev->prev.canvas.x);

if(tile_info[1][1].curr_x < -(X_RESOLUTION-80))
{
    set_map_main_tile_info(current_center.x_info+1, current_center.y_info);
    update_main_page();
}

```

```

        for(int i=0; i<MAP_TILE_X ; i++)
    {
        for(int j=0; j<MAP_TILE_Y ; j++)
        {
            evas_object_move(main_page[i][j], tile_info[i][j].curr_x+X_RESOLUTION,
tile_info[i][j].curr_y);
            tile_info[i][j].curr_x = tile_info[i][j].curr_x+X_RESOLUTION;
        }
    }
}

// y point check
if(ev->cur.canvas.y > ev->prev.canvas.y) // y++
{
    if(tile_info[1][1].curr_y > 0)
    {
        for(int i=0 ; i<MAP_TILE_Y ; i++)
        {
            if(
                all_tiles[current_center.x_info-(1-
i)][current_center.y_info+2].download_request)
            {
                double lati_of_center =
all_tiles[current_center.x_info][current_center.y_info].lati_of_center;
                double longti_of_center =
all_tiles[current_center.x_info][current_center.y_info].longti_of_center;

                lati_of_center = next_lati_value(lati_of_center, Y_RESOLUTION*(-1),
current_zoom_level);
                lati_of_center = next_lati_value(lati_of_center, Y_RESOLUTION*(-1),
current_zoom_level);
                longti_of_center = next_longti_value(longti_of_center,
X_RESOLUTION*(i-1), current_zoom_level);
            }
        }
    }
}

```

```

        request_map_download(current_center.x_info-(1-i),
current_center.y_info+2, lati_of_center, longti_of_center);
    }
}

for(int i=0; i<MAP_TILE_X ; i++)
{
    for(int j=0; j<MAP_TILE_Y ; j++)
    {
        tile_info[i][j].curr_y = tile_info[i][j].curr_y+(ev->cur.canvas.y - ev->prev.canvas.y);
        evas_object_move(main_page[i][j], tile_info[i][j].curr_x, tile_info[i][j].curr_y);
    }
}
y_moved = y_moved + (ev->cur.canvas.y - ev->prev.canvas.y);

if(tile_info[1][1].curr_y > Y_RESOLUTION)
{
    set_map_main_tile_info(current_center.x_info, current_center.y_info+1);
    update_main_page();

    for(int i=0; i<MAP_TILE_X ; i++)
    {
        for(int j=0; j<MAP_TILE_Y ; j++)
        {
            evas_object_move(main_page[i][j], tile_info[i][j].curr_x,
tile_info[i][j].curr_y-Y_RESOLUTION);
            tile_info[i][j].curr_y = tile_info[i][j].curr_y-Y_RESOLUTION;
        }
    }
}
else if(ev->cur.canvas.y < ev->prev.canvas.y) // y--
{

```

```

if(tile_info[1][1].curr_y < 0)
{
    for(int i=0 ; i<MAP_TILE_Y ; i++)
    {
        if(!all_tiles[current_center.x_info-(1-i)][current_center.y_info-2].download_request)
        {
            double lati_of_center = all_tiles[current_center.x_info][current_center.y_info].lati_of_center;
            double longti_of_center = all_tiles[current_center.x_info][current_center.y_info].longti_of_center;

            lati_of_center = next_lati_value(lati_of_center, Y_RESOLUTION, current_zoom_level);
            lati_of_center = next_lati_value(lati_of_center, Y_RESOLUTION, current_zoom_level);
            longti_of_center = next_longti_value(longti_of_center, X_RESOLUTION*(i-1), current_zoom_level);

            request_map_download(current_center.x_info-(1-i), current_center.y_info-2, lati_of_center, longti_of_center);
        }
    }
}

for(int i=0; i<MAP_TILE_X ; i++)
{
    for(int j=0; j<MAP_TILE_Y ; j++)
    {
        tile_info[i][j].curr_y = tile_info[i][j].curr_y+(ev->cur.canvas.y - ev->prev.canvas.y);
        evas_object_move(main_page[i][j], tile_info[i][j].curr_x, tile_info[i][j].curr_y);
    }
}
y_moved = y_moved + (ev->cur.canvas.y - ev->prev.canvas.y);

```

```

if(tile_info[1][1].curr_y < -(Y_RESOLUTION-200))
{
    set_map_main_tile_info(current_center.x_info, current_center.y_info-1);
    update_main_page();

    for(int i=0; i<MAP_TILE_X ; i++)
    {
        for(int j=0; j<MAP_TILE_Y ; j++)
        {
            evas_object_move(main_page[i][j], tile_info[i][j].curr_x,
tile_info[i][j].curr_y+Y_RESOLUTION);
            tile_info[i][j].curr_y = tile_info[i][j].curr_y+Y_RESOLUTION;
        }
    }
}

void mouse_up_cb(void *data, Evas *e , Evas_Object *obj , void *event_info)
{
    Evas_Event_Mouse_Move *ev;
    ev = (Evas_Event_Mouse_Move *)event_info;

    //appdata_s *ad = data;
    mapdata_s *md = data;
    md->mouse_down = EINA_FALSE;
}

void mouse_down_cb(void *data, Evas *e , Evas_Object *obj , void *event_info)
{
    Evas_Event_Mouse_Move *ev;
    ev = (Evas_Event_Mouse_Move *)event_info;
}

```

```

//appdata_s *ad = data;
mapdata_s *md = data;
md->mouse_down = EINA_TRUE;
}

void update_main_page(void)
{
    for(int i=0; i<MAP_TILE_X ; i++)
    {
        for(int j=0; j<MAP_TILE_Y ; j++)
        {
            if(main_page[i][j] != NULL)
            {
                evas_object_del(main_page[i][j]);
                main_page[i][j] = NULL;
            }

            main_page[i][j] = evas_object_image_filled_add(m_canvas);

            evas_object_image_file_set(main_page[i][j], tile_info[i][j].file_name, NULL);
            evas_object_move(main_page[i][j], tile_info[i][j].curr_x, tile_info[i][j].curr_y);
            evas_object_resize(main_page[i][j], X_RESOLUTION, Y_RESOLUTION);
            evas_object_show(main_page[i][j]);
            evas_object_raise (m_btn1);
            evas_object_raise (m_btn2);

            evas_object_event_callback_add(main_page[i][j],
EVAS_CALLBACK_MOUSE_MOVE, mouse_move_cb, m_md);
            evas_object_event_callback_add(main_page[i][j],
EVAS_CALLBACK_MOUSE_DOWN, mouse_down_cb, m_md);
            evas_object_event_callback_add(main_page[i][j],    EVAS_CALLBACK_MOUSE_UP,
mouse_up_cb, m_md);
        }
    }
}

```

```

void map_dload_thread(void *data, Ecore_Thread *thread)
{
    int dload_success=0;

    while (1)
    {
        while(map_dload_q_idx && !do_not_update_map)
        {

            char dload_url[MAX_URL_LEN] = {0,};
            char dload_filename[MAX_TMP_FILENAME_LEN] = {0,};
            index_of_all_tile_s index;

            eina_lock_take(&set_info_mutex);
            map_dload_q_idx--;
            memcpy(dload_url, &map_dload_q[map_dload_q_idx].url[0], MAX_URL_LEN);
            memcpy(dload_filename,     &map_dload_q[map_dload_q_idx].filename[0],
MAX_TMP_FILENAME_LEN);
            index.x = map_dload_q[map_dload_q_idx].index.x;
            index.y = map_dload_q[map_dload_q_idx].index.y;
            eina_lock_release(&set_info_mutex);

            dload_success = 0;
            do
            {
                dload_success = map_download(dload_url, dload_filename);
            } while(dload_success==0);

            all_tiles[index.x][index.y].download_ok = 1;
            set_map_main_tile_info(current_center.x_info, current_center.y_info);
            update_main_page();

            //ecore_thread_feedback(thread, &index);
        }
    }
}

```

```

}

void thread_feedback(void *data, Ecore_Thread *thread, void *msg_data)
{
    //TODO
}

void thread_end(void *data, Ecore_Thread *thread)
{
    //TODO
}

void thread_cancel(void *data, Ecore_Thread *thread)
{
    //TODO
}

void loc_state_changed_cb(location_service_state_e state, void *user_data)
{
    if(state == LOCATIONS_SERVICE_ENABLED)
        location_initialized = 1;
}

int make_start_url(double lati, double longti)
{
    char buf[GEO_INFO_STR_NUM];
    map_start_lati_of_center = lati;
    map_start_longti_of_center = longti;

    if(make_new_url(ZOOM_CHANGED, 0, NULL, 0, 0) != URL_CHANGED)
        return URL_NOT_CHANGED;

    memset(buf, 0, GEO_INFO_STR_NUM);
    snprintf(buf, sizeof(buf), "%f,%f", map_start_lati_of_center, map_start_longti_of_center);
    if(make_new_url(CENTER_NAME_CHANGED, 0, buf, 0, 0) != URL_CHANGED)

```

```

    return URL_NOT_CHANGED;

    return URL_CHANGED;
}

void arrange_start_main_page(void)
{
    map_dload_q_idx = 0;
    do_not_update_map = 1;

    x_moved = 0;
    y_moved = 0;

    current_center.x_info = ALL_TILE_USABLE_X/2;
    current_center.y_info = ALL_TILE_USABLE_Y/2;

    for(int i=0; i<ALL_TILE_USABLE_X ; i++)
    {
        for(int j=0; j<ALL_TILE_USABLE_Y ; j++)
        {
            all_tiles[i][j].download_ok = 0;
            all_tiles[i][j].download_request = 0;
            memset(all_tiles[i][j].file_name, 0, MAX_TMP_FILENAME_LEN);
        }
    }

    for(int i=0; i<MAP_TILE_X ; i++)
    {
        for(int j=0; j<MAP_TILE_Y ; j++)
        {
            tile_info[i][j].curr_x = tile_info_init_information[i][j].curr_x;
            tile_info[i][j].curr_y = tile_info_init_information[i][j].curr_y;

            if(i==1 && j==1) continue;
            request_map_download(current_center.x_info-(1-i), current_center.y_info-(1-j),

```

```

        next_lati_value(map_start_lati_of_center,           Y_RESOLUTION*(1-j),
current_zoom_level),
        next_longti_value(map_start_longti_of_center,       X_RESOLUTION*(i-1),
current_zoom_level));
    }
}

//download ceter map first for visual effect
request_map_download(current_center.x_info,           current_center.y_info,
map_start_lati_of_center, map_start_longti_of_center);
do_not_update_map = 0;

set_map_main_tile_info(current_center.x_info, current_center.y_info);
}

void map_zoom_in()
{
if(make_new_url(ZOOM_CHANGED, 1, NULL, 0, 0) == URL_CHANGED)
{
    map_start_lati_of_center = next_lati_value(map_start_lati_of_center, y_moved*(-1),
current_zoom_level-1);
    map_start_longti_of_center = next_longti_value(map_start_longti_of_center, -x_moved,
current_zoom_level-1);

    arrange_start_main_page();
}
}

void map_zoom_out()
{
if(make_new_url(ZOOM_CHANGED, -1, NULL, 0, 0) == URL_CHANGED)
{
    map_start_lati_of_center = next_lati_value(map_start_lati_of_center, y_moved*(-1),
current_zoom_level+1);
    map_start_longti_of_center = next_longti_value(map_start_longti_of_center, -x_moved,
current_zoom_level+1);
}
}

```

```

        arrange_start_main_page();

    }

}

static size_t
WriteMemoryCallback(void *contents, size_t size, size_t nmemb, void *userp)
{
    size_t realsize = size * nmemb;
    struct MemoryStruct *mem = (struct MemoryStruct *)userp;

    mem->memory = realloc(mem->memory, mem->size + realsize + 1);
    if(mem->memory == NULL) {
        /* out of memory! */
        printf("not enough memory (realloc returned NULL)\n");
        return 0;
    }

    memcpy(&(mem->memory[mem->size]), contents, realsize);
    mem->size += realsize;
    mem->memory[mem->size] = 0;

    return realsize;
}

int get_geometry_info_of_place(char* url, double* start_latitude, double* start_longitude)
{
    CURL *curl_handle;
    MemoryStruct_s chunk;
    int ret_val=0;
    char* search_index = NULL;
    char* pos = NULL;
    char* pos_lati_start = NULL;
    char* pos_lati_end = NULL;
    char* pos_longti_start = NULL;

```

```

char* pos_longti_end = NULL;
char tmp_str_lati[GEO_INFO_STR_NUM]={0,};
tmp_str_longti[GEO_INFO_STR_NUM]={0,};

double tmp_lati_sum=0, tmp_longti_sum=0;

chunk.memory = malloc(1); /* will be grown as needed by the realloc above */
chunk.size = 0; /* no data at this point */

curl_global_init(CURL_GLOBAL_ALL);

/* init the curl session */
curl_handle = curl_easy_init();

/* set URL to get */
curl_easy_setopt(curl_handle, CURLOPT_URL, url);
curl_easy_setopt(curl_handle, CURLOPT_SSL_VERIFYPeer, 0L);

/* no progress meter please */
//curl_easy_setopt(curl_handle, CURLOPT_FOLLOWLOCATION, 1L);

/* send all data to this function */
curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, WriteMemoryCallback);

/* we want the headers to this file handle */
curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, (void *)&chunk);

/* some servers don't like requests that are made without a user-agent
   field, so we provide one */
curl_easy_setopt(curl_handle, CURLOPT_USERAGENT, "libcurl-agent/1.0");

/*
 *      * Notice here that if you want the actual data sent anywhere else but
 *      *      * stdout, you should consider using the CURLOPT_WRITEDATA option.  */
/* get it! */

```

```

ret_val = curl_easy_perform(curl_handle);

memset(temp_place_info, 0, PLACE_INFO_MAX);
memcpy(temp_place_info, chunk.memory, chunk.size);

/* cleanup curl stuff */
curl_easy_cleanup(curl_handle);

if(chunk.memory)
    free(chunk.memory);

/* we're done with libcurl, so clean it up */
curl_global_cleanup();

place_searched_num = 0;
search_index = temp_place_info;
while(1)
{
    pos = strstr(search_index, "<location>");
    if(pos != NULL)// detected one more
    {
        pos_lati_start = strstr((char*)pos, "<lat>");
        pos_lati_end = strstr((char*)pos, "</lat>");
        pos_longti_start = strstr((char*)pos, "<lng>");
        pos_longti_end = strstr((char*)pos, "</lng>");

        memset(tmp_str_lati, 0, GEO_INFO_STR_NUM);
        memcpy(tmp_str_lati, (char*)(pos_lati_start+5), pos_lati_end-(pos_lati_start+5));
        place_search_list[place_searched_num].str_lati = atof(tmp_str_lati);

        memset(tmp_str_longti, 0, GEO_INFO_STR_NUM);
        memcpy(tmp_str_longti, (char*)(pos_longti_start+5), pos_longti_end-(pos_longti_start+5));
        place_search_list[place_searched_num].str_longti = atof(tmp_str_longti);
    }
}

```

```

        place_searched_num++;
        search_index = pos+1;//just for next search
    }
    else
    {
        break; //couldn't find anymore.
    }
}

for(int i=0 ; i<place_searched_num ; i++)
{
    tmp_lat_i_sum = tmp_lat_i_sum + place_search_list[i].str_lat;
    tmp_longti_sum = tmp_longti_sum + place_search_list[i].str_longti;
}

*start_latitude = place_search_list[0].str_lat;
*start_longitude = place_search_list[0].str_longti;

return ret_val;
}

//handle key event for search
void key_down_cb(void *data, Evas *evas, Evas_Object *obj, void *event_info)
{
    const int MAX_CUR = 20;
    static char buf[50];
    static int cur = 0;
    double latitude, longitude;
    static char geometry_buf[50]={0,};

    Evas_Event_Key_Down *ev = event_info;
    Evas_Object *input = data;
    char tmp[50];

    if (cur == 0) snprintf(buf, sizeof(buf), "%0");
}

```

```

if (!strcmp(ev->keyname, "Return"))
{
    evas_object_text_text_set(input, "");
    cur = 0;
    for(int i =0 ; i<strlen(buf) ; i++)
    {
        if(buf[i]==' ') buf[i]='; // google api can't recognize ' '
    }
    if(make_new_url(PLACE_INFO, 0, buf, 0, 0) == URL_CHANGED)//make url for query
of place info.

    {
        if(!get_geometry_info_of_place(place_api_url, &latitude, &longitude) // get
place info data using query url and parse it.

        {
            memset(geometry_buf, 0, 50);
            sprintf(geometry_buf, sizeof(geometry_buf), "%f,%f", latitude, longitude);
            make_new_url(DELITE_MARKER, 0, NULL, 0, 0); // delete all previous markers
            if((make_new_url(CENTER_NAME_CHANGED, 0, geometry_buf, 0, 0) ==
URL_CHANGED)
                && (make_new_url(ADD_NEW_MULTI_MARKER, 0, NULL, 0, 0) ==
URL_CHANGED))
            {
                map_start_lati_of_center = latitude;
                map_start_longti_of_center = longitude;
                arrange_start_main_page();
                update_main_page();
            }
        }
    }
    return;
}

if (!strcmp(ev->keyname, "BackSpace"))
{

```

```

snprintf(tmp, strlen(buf), "%s", buf);
evas_object_text_text_set(input, tmp);
strcpy(buf, tmp);
cur--;
return;
}

if (cur >= MAX_CUR) return;

if(!strcmp(ev->keyname, "space"))
{
    snprintf(tmp, sizeof(tmp), "%s%s", buf, " "); // replace ' ' to ' '
    evas_object_text_text_set(input, tmp);
    cur++;
    strcpy(buf, tmp);
}

else if(!strcmp(ev->keyname, "comma"))
{
    snprintf(tmp, sizeof(tmp), "%s%s", buf, ",");
    evas_object_text_text_set(input, tmp);
    cur++;
    strcpy(buf, tmp);
}

else if(!strcmp(ev->keyname, "Caps_Lock"))
|| !strcmp(ev->keyname, "Shift_L")
|| !strcmp(ev->keyname, "Num_Lock")
|| !strcmp(ev->keyname, "Left")
|| !strcmp(ev->keyname, "Right")
|| !strcmp(ev->keyname, "Up")
|| !strcmp(ev->keyname, "Down"))

{
    // to be ignored...
}

else
{

```

```

snprintf(tmp, sizeof(tmp), "%s%s", buf, ev->keyname);
evas_object_text_text_set(input, tmp);
cur++;
strcpy(buf, tmp);
}

}

void go_to_current_geometry(void)
{
    double altitude, climb, direction, speed;
    double horizontal, vertical;location_accuracy_level_e level;time_t timestamp;
    char buf[100];

    if(!location_initialized)
        return;

    location_manager_get_last_location(l_manager, &altitude, &curr_user_lat,
&curr_user_longti,
&climb, &direction, &speed, &level, &horizontal,
&vertical, &timestamp);

    memset(buf, 0, 100);
    snprintf(buf, sizeof(buf), "%f,%f", curr_user_lat, curr_user_longti);
    if(make_new_url(CENTER_NAME_CHANGED, 0, buf, 0, 0) != URL_CHANGED)
        return;
    make_new_url(DELITE_MARKER, 0, NULL, 0, 0);
    if(make_new_url(ADD_NEW_MARKER, 0, buf, curr_user_longti, curr_user_lat) != URL_CHANGED)
        return;

    map_start_lati_of_center = curr_user_lat;
    map_start_longti_of_center = curr_user_longti;

    arrange_start_main_page();
}

```

```

    get_geometry_timer = ecore_timer_add(1, get_changed_location, NULL);
}

Eina_Bool get_changed_location(void *data)
{
    double altitude, latitude, longitude, climb, direction, speed;
    double horizontal, vertical; location_accuracy_level_e level; time_t timestamp;

    if(!location_initialized)
        return ECORE_CALLBACK_RENEW;

    location_manager_get_last_location(l_manager, &altitude, &latitude, &longitude,
                                       &climb, &direction, &speed, &level, &horizontal,
                                       &vertical, &timestamp);

    if((latitude != curr_user_lat) || (longitude != curr_user_long))
    {
        ecore_timer_del(get_geometry_timer);
        go_to_current_geometry();
        return ECORE_CALLBACK_CANCEL;
    }
    else
    {
        return ECORE_CALLBACK_RENEW;
    }
}

//go to current user's point
void current_btn_cb(void *data, Evas *e , Evas_Object *obj , void *event_info)
{
    go_to_current_geometry();
}

static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{

```

```

int btn_num = (int)data;
//dlog_print(DLOG_ERROR, "tag", "clicked event on Button:%d", btn_num);

switch( btn_num ) {
case 1 :
    map_zoom_in();
    break;
case 2 :
    map_zoom_out();
    break;
}
}

void create_map(Evas_Object *win, double lati, double longti)
{
    m_md = (mapdata_s*)malloc( sizeof( mapdata_s ) );

    /* Button-1 */
    m_btn1 = elm_button_add(win);
    elm_object_text_set(m_btn1, "+");
    evas_object_move(m_btn1, 20, 20);
    evas_object_resize(m_btn1, 50, 50);
    evas_object_smart_callback_add(m_btn1, "clicked", btn_clicked_cb, (void *)1);
    evas_object_show(m_btn1);

    /* Button-2 */
    m_btn2 = elm_button_add(win);
    elm_object_text_set(m_btn2, "-");
    evas_object_move(m_btn2, 90, 20);
    evas_object_resize(m_btn2, 50, 50);
    evas_object_smart_callback_add(m_btn2, "clicked", btn_clicked_cb, (void *)2);
    evas_object_show(m_btn2);

    /* Canvas */
    m_canvas = evas_object_evas_get(win);

```

```

char *res_path = app_get_resource_path();
if (res_path) {
    snprintf(m_icon_path, PATH_MAX, "%s%s", res_path, "images");
    free(res_path);
}

/* Thread */
ecore_thread_feedback_run(map_dload_thread,      thread_feedback,      thread_end,
thread_cancel, NULL, EINA_TRUE);
eina_lock_new(&set_info_mutex);

/* LocationManager */
location_manager_create(LOCATIONS_METHOD_GPS, &l_manager);
location_manager_set_service_state_changed_cb(l_manager,      loc_state_changed_cb,
NULL);
location_manager_start(l_manager);

mkdir(TMP_DIR, 0755);

if( make_start_url(lati, longti) != URL_CHANGED ) //let's make start url to load
return;

arrange_start_main_page();
}

#endif /* MAPVIEW_H_ */

```

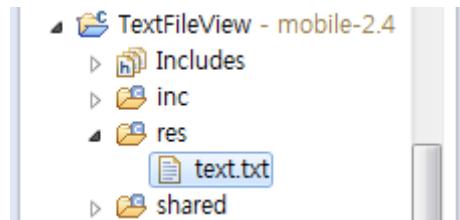
58. 텍스트 파일 읽고 쓰기

파일에서 텍스트 데이터를 읽고 쓸때는 FILE 구조체를 사용하면 됩니다.
/res 폴더에 있는 파일은 Read는 가능하지만 Write는 안됩니다. /data
폴더에 있는 파일은 Read와 Write 모두 가능합니다.

1) 텍스트 파일 Read

/res 폴더에 있는 텍스트 파일을 읽어서 화면에 출력해 보겠습니다.

새로운 소스 프로젝트를 생성하고 Project name을 TextFileDialog 로
지정합니다. 소스 프로젝트가 생성되었으면 부록 /etc 폴더에 있는
text.txt 파일을 소스 프로젝트 /res 폴더로 복사합니다.



ext.txt 파일에는 아래와 같이 세계 여러 나라 언어로 인사말이 저장되어 있습니다.

Good morning

早上好

Hyvää Huomenta

Bonjour

Guten Morgen

Jó reggelt kívánok

Buon giorno

おはようございます。

안녕하세요

Bună dimineața!

Buenos Días.

Günaydın

Xin chào

Здравствуйте

src 폴더에 소스파일(~.c)을 열고 appdata 구조체에 변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    //Evas_Object *label;
    Evas_Object *entry;
} appdata_s;
```

Entry 위젯 변수를 추가하였습니다.

create_base_gui() 함수 위에 2개의 새로운 함수를 생성합니다.

```
static void
my_table_pack(Evas_Object *table, Evas_Object *child, int col, int row, int spanx, int
spany,
              double h_expand, double v_expand, double h_align, double v_align)
{
    /* Create a frame around the child, for padding */
    Evas_Object *frame = elm_frame_add(table);
    elm_object_style_set(frame, "pad_small");

    evas_object_size_hint_weight_set(frame, h_expand, v_expand);
```

```

evas_object_size_hint_align_set(frame, h_align, v_align);

/* place child in its box */
{
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    elm_object_content_set(frame, child);
    evas_object_show(child);
}

elm_table_pack(table, frame, col, row, spanx, spany);
evas_object_show(frame);
}

static Evas_Object *
my_button_add(Evas_Object *parent, const char *text, Evas_Smart_Cb cb, void *cb_data)
{
    Evas_Object *btn;

    btn = elm_button_add(parent);
    elm_object_text_set(btn, text);
    evas_object_smart_callback_add(btn, "clicked", cb, cb_data);

    return btn;
}

```

my_table_pack() 은 Table에 위젯을 추가하는 함수입니다.

my_button_add() 는 Button 위젯을 생성하는 함수입니다.

그런 다음 create_base_gui() 함수로 이동해서 Frame, Table, Button 그리고 Entry 위젯 생성 코드를 추가합니다. Label 위젯 생성 코드는

주석 처리합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
    Evas_Object *tbl, *btn, *frame;

    /* Frame */
    frame = elm_frame_add(ad->win);
    elm_object_style_set(frame, "pad_medium");
    elm_object_content_set(ad->conform, frame);
    evas_object_size_hint_weight_set(frame, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(frame, EVAS_HINT_FILL, EVAS_HINT_FILL);
    evas_object_show(frame);

    /* Container: standard table */
    tbl = elm_table_add(ad->win);
    evas_object_size_hint_weight_set(tbl, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(tbl, EVAS_HINT_FILL, EVAS_HINT_FILL);
    elm_object_content_set(frame, tbl);
    evas_object_show(tbl);

    {
        /* Button-1 */
        btn = my_button_add(ad->conform, "Read", btn_read_cb, ad);
```

```

        my_table_pack(tbl, btn, 0, 0, 1, 1, EVAS_HINT_EXPAND, 0.0,
EVAS_HINT_FILL, EVAS_HINT_FILL);

        /* Entry */
        ad->entry = elm_entry_add(ad->conform);
        elm_entry_scrollable_set(ad->entry, EINA_TRUE);
        elm_object_signal_emit(ad->entry, "elm,state,scroll,enabled", "");
        elm_object_text_set(ad->entry, "Please press <b>Read</> button");
        my_table_pack(tbl, ad->entry, 0, 1, 2, 1, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND, EVAS_HINT_FILL, EVAS_HINT_FILL);
    }

}

/* Show window after base gui is set up */
evas_object_show(ad->win);
}

```

create_base_gui() 함수 위에 새로운 함수 3개를 추가합니다.

```

static void
app_get_resource(const char *res_file_in, char *res_path_out, int res_path_max)
{
    char *res_path = app_get_resource_path();
    if (res_path) {
        snprintf(res_path_out, res_path_max, "%s%s", res_path, res_file_in);
        free(res_path);
    }
}

static char*
read_file(const char* filepath)
{
    FILE *fp = fopen(filepath, "r");

```

```

if (fp == NULL)
    return NULL;
fseek(fp, 0, SEEK_END);
int bufsize = ftell(fp);
rewind(fp);
if (bufsize < 1)
    return NULL;

char *buf = malloc(sizeof(char) * (bufsize));
memset(buf, '\0', sizeof(buf));
char str[200];

while(fgets(str, 200, fp) != NULL) {
    dlog_print(DLOG_ERROR, "tag", "%s", str);
    sprintf(buf + strlen(buf), "%s", str);
}

fclose(fp);
return buf;
}

static void
btn_read_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char filepath[PATH_MAX] = { 0, };
    char *buf = NULL;
    app_get_resource("text.txt", filepath, PATH_MAX);
    buf = read_file(filepath);

    elm_object_text_set(ad->entry, buf);
}

```

app_get_resource() 는 /res 폴더에 저장된 파일의 절대경로를 구해서

반환하는 함수입니다.

`read_file()` 는 텍스트 파일의 내용을 읽어서 반환하는 함수입니다.

`fopen (char *, char *)` 은 파일의 핸들을 반환하는 API입니다. 1번째 파라미터에는 파일 경로를 전달하고, 2번째 파라미터에는 파일 접근 모드를 지정합니다. "r" 이면 Read 전용이고, "w" 이면 Write 전용을 의미합니다.

`fseek (FILE *, int, int)` 는 파일 스트림의 특정 위치로 이동하는 API입니다. 1번째 파라미터는 파일 스트림, 2번째는 이동 바이트 수, 3번째는 시작 위치입니다. SEEK_SET 는 파일 시작, SEEK_CUR 는 현재 위치, SEEK_END 는 파일의 끝을 의미합니다.

`ftell (FILE *)` 는 파일 스트림의 현재 위치를 바이트 수로 반환하는 API입니다. 끝부분에 있을 때는 파일 크기를 반환하게 됩니다.

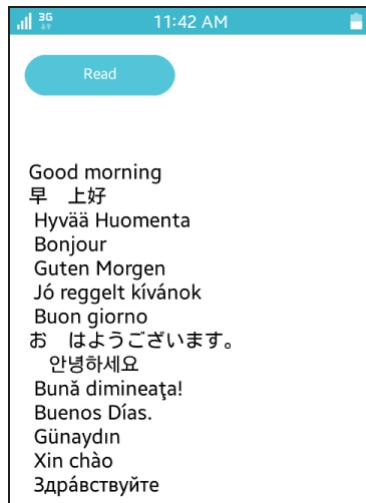
`rewind (FILE *)` 는 파일 스트림 위치를 처음으로 되돌리는 API입니다.

`fgets (char *, int, FILE *)` 는 파일에서 텍스트 데이터를 읽는 API입니다. 2번째 파라미터에 최대 문자열 길이, 3번째 파라미터에 파일 스트림을 전달하면 1번째 파라미터에서 문자열 데이터를 반환해 줍니다.

`fclose (FILE *)` 는 파일 스트림을 닫는 API 입니다.

`btn_read_cb()` 는 Button 콜백 함수입니다. 텍스트 파일 경로를 구하고 해당 파일의 내용을 읽어서 Entry 위젯에 입력합니다.

예제를 빌드하고 실행시켜 봅시다. Read 버튼을 누르면 각 언어별로 인사말이 Entry 위젯에 출력됩니다.



2) 텍스트 파일 Write

Button을 하나 더 추가해서 수정된 텍스트를 파일로 저장하는 기능을 구현 하겠습니다. /res 폴더에는 파일 Write가 허용되지 않습니다. 그래서 /data 폴더에 저장해야 합니다. `create_base_gui()` 함수에 2번째 Button 생성 코드를 추가합니다.

```
{
    /* Button-1 */
    btn = my_button_add(ad->conform, "Read", btn_read_cb, ad);
    my_table_pack(tbl, btn, 0, 0, 1, 1, EVAS_HINT_EXPAND, 0.0, EVAS_HINT_FILL,
EVAS_HINT_FILL);

    /* Button-2 */
    btn = my_button_add(ad->conform, "Write", btn_write_cb, ad);
    my_table_pack(tbl, btn, 1, 0, 1, 1, EVAS_HINT_EXPAND, 0.0,
EVAS_HINT_FILL, EVAS_HINT_FILL);

    /* Entry */
```

```

        ad->entry = elm_entry_add(ad->conform);
        elm_entry_scrollable_set(ad->entry, EINA_TRUE);
        elm_object_signal_emit(ad->entry, "elm,state,scroll,enabled", "");
        elm_object_text_set(ad->entry, "Please press <b>Read</> button");
        my_table_pack(tbl, ad->entry, 0, 1, 2, 1, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND, EVAS_HINT_FILL, EVAS_HINT_FILL);
    }
}

```

텍스트를 파일에 저장하기 위해서 `create_base_gui()` 함수 위에 새로운 함수 3개를 추가합니다.

```

static void
app_get_data(const char *res_file_in, char *res_path_out, int res_path_max)
{
    char *res_path = app_get_data_path();
    if (res_path) {
        snprintf(res_path_out, res_path_max, "%s%s", res_path, res_file_in);
        free(res_path);
    }
}

static char*
write_file(const char* filepath, const char* buf)
{
    FILE *fp;
    fp = fopen(filepath, "w");
    fputs(buf, fp);
    fclose(fp);
}

static void
btn_write_cb(void *data, Evas_Object *obj, void *event_info)

```

```
{  
    appdata_s *ad = data;  
    char* buf = elm_entry_entry_get(ad->entry);  
  
    char filepath[PATH_MAX] = { 0, };  
    app_get_data("text.txt", filepath, PATH_MAX);  
    write_file(filepath, buf);  
}
```

app_get_data() 는 /data 폴더에 존재하는 파일의 절대 경로를 반환하는 함수입니다.

app_get_data_path() 는 /data 폴더의 절대경로를 반환하는 API 입니다.

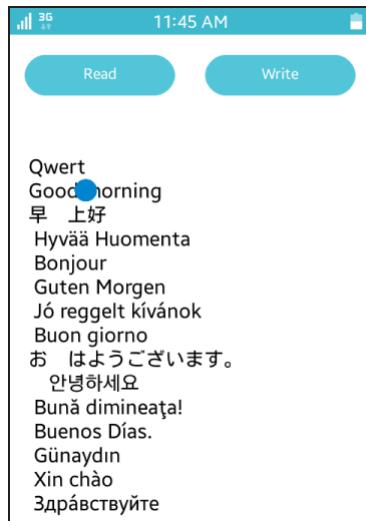
write_file() 은 파일에 텍스트 데이터를 저장하는 함수입니다.

fputs (char *, FILE *) 는 파일 스트림에 텍스트 데이터를 저장하는 API 입니다.

btn_write_cb() 는 2번째 Button의 콜백 함수입니다. /data 폴더의 절대경로를 구해서 Entry에 입력된 텍스트 데이터를 파일에 저장합니다.

예제를 다시 실행시켜 봅시다. Read 버튼을 눌러서 파일을 불러오고, Entry의 내용을 수정한 다음, Write 버튼을 눌러봅시다.

이제 수정된 내용이 /data 폴더에 저장되었습니다.



3) /data 폴더 파일 Read

안타깝게도 앱을 종료했다가 다시 실행하면 파일의 내용이 변경되지 않았습니다. 저장된 파일은 /data 폴더에 저장되었는데, /res 폴더에서 파일을 읽어오기 때문입니다. 파일을 읽을 때 /data 폴더에서 먼저 파일을 찾고, 만약 파일이 없으면 /res 폴더에서 찾도록 기능을 변경해 보겠습니다.

btn_read_cb() 함수의 내용을 아래와 같이 수정합니다. 이 함수의 위치는 app_get_data() 보다 아래쪽에 위치해야 합니다.

```
static void  
app_get_data(const char *res_file_in, char *res_path_out, int res_path_max)  
{  
    char *res_path = app_get_data_path();  
    if (res_path) {  
        sprintf(res_path_out, res_path_max, "%s%s", res_path, res_file_in);  
        free(res_path);  
    }  
}
```

```

    }

}

static void
btn_read_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char filepath[PATH_MAX] = { 0, };
    app_get_data("text.txt", filepath, PATH_MAX);
    // Read file in /data folder
    char *buf = NULL;
    buf = read_file(filepath);
    // If doesn't exist file in /data folder, read file in /res folder
    if( buf == NULL ) {
        app_get_resource("text.txt", filepath, PATH_MAX);
        buf = read_file(filepath);
    }

    elm_object_text_set(ad->entry, buf);
}

```

/data 폴더에서 먼저 파일을 읽습니다. 만약 데이터가 존재하지 않으면 /res 폴더에서 읽는 것입니다.

다시 예제를 실행시켜 봅시다. Read 버튼을 눌러서 파일 불러오고, Entry 위젯 내용을 수정한 다음, Write 버튼을 눌러서 저장합니다.

앱을 종료했다가(Home 버튼을 오래 눌러서 앱 목록이 나타나면 Clear all을 클릭) 다시 실행합니다. Read 버튼을 눌러서 수정된 내용이 보이면 완성입니다.

4) 관련 API

FILE *fopen (char *, char *) : 파일의 핸들을 반환하는 API. 1번째 파라미터에는 파일 경로를 전달하고, 2번째 파라미터에는 파일 접근 모드를 지정합니다. "r" 이면 Read 전용이고, "w" 이면 Write 전용을 의미합니다.

int fseek (FILE *_stream, long int _off, int _whence) : 파일 스트림의 특정 위치로 이동하는 API. 1번째 파라미터는 파일 스트림, 2번째는 이동 바이트 수, 3번째는 시작 위치입니다. SEEK_SET 는 파일 시작, SEEK_CUR 는 현재 위치, SEEK_END 는 파일의 끝을 의미합니다.

long int ftell (FILE *_stream) : 파일 스트림의 현재 위치를 바이트 수로 반환하는 API. 끝부분에 있을 때는 파일 크기를 반환하게 됩니다.

void rewind (FILE *_stream) : 파일 스트림 위치를 처음으로 되돌리는 API.

char *fgets (char *_s, int _n, FILE *_stream) : 파일에서 텍스트 데이터를 읽는 API. 2번째 파라미터에 최대 문자열 길이, 3번째 파라미터에 파일 스트림을 전달하면 1번째 파라미터에서 문자열 데이터를 반환해 줍니다.

int fclose (FILE *_stream) : 파일 스트림을 닫는 API.

char *app_get_data_path(void) : /data 폴더의 절대경로를 반환하는 API.

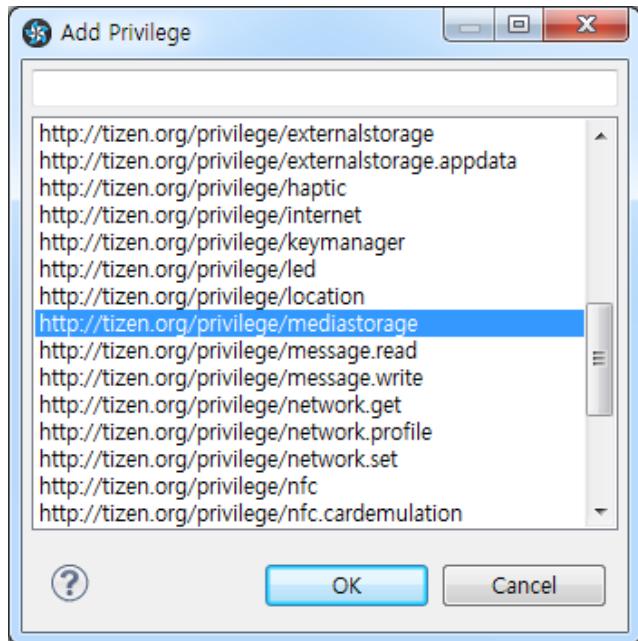
int fputs (char *_s, FILE *_stream) : 파일 스트림에 텍스트 데이터를 저장하는 API.

59. 파일 목록 구하기

Astro 같은 파일 관리 앱은 수년간 필수앱의 자리를 지키고 있습니다. 파일 관리 앱 뿐만 아니라 이미지 뷰어 혹은 오디오 플레이어 같은 컨텐츠 재생 앱에서도 파일 목록을 구하는 기능이 필수적입니다. 이번 예제에서는 파일과 폴더 목록을 List 위젯에 추가하고 항목을 선택해서 폴더 경로를 이동하는 간단한 파일 관리자 앱을 제작해 보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 fileList 으로 지정합니다. 메모리에 저장된 파일 목록을 구하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/mediastorage> 을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.

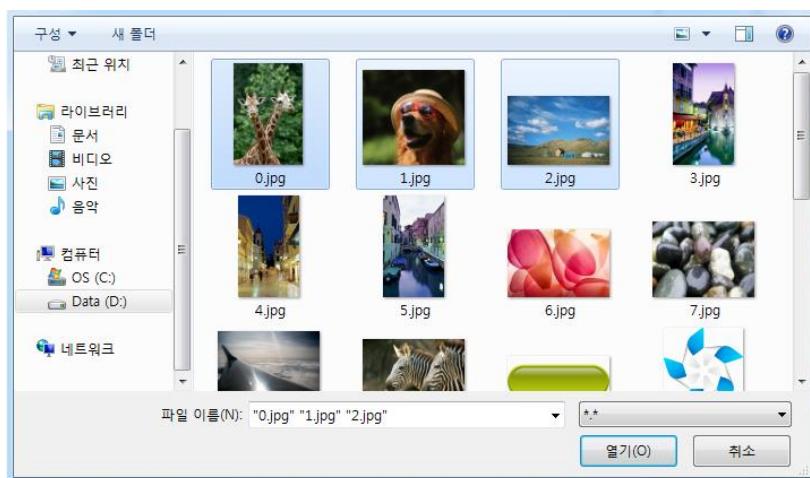
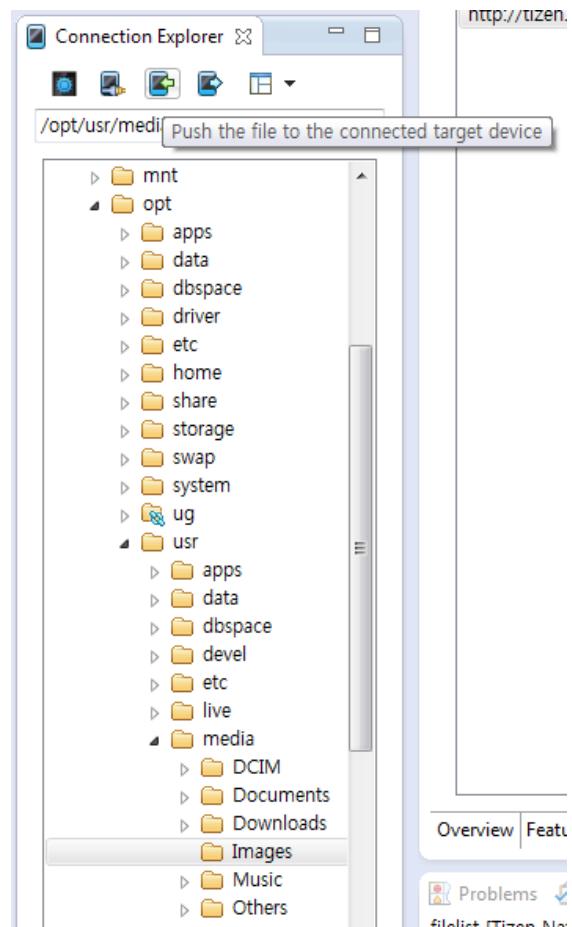


저장하고 아래쪽 탭버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

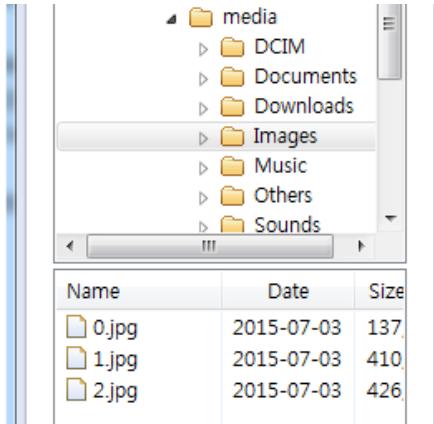
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns="http://tizen.org/ns/packages" api-version="2.3"
package="org.example.filelist" version="1.0.0">
    <profile name="mobile" />
    <ui-application appid="org.example.filelist" exec="filelist" type="capp"
multiple="false" taskmanage="true" nodisplay="false">
        <icon>filelist.png</icon>
        <label>filelist</label>
    </ui-application>
    <privileges>
        <b><privilege>http://tizen.org/privilege/mediastorage</privilege></b>
    </privileges>
</manifest>
```

에뮬레이터의 공유메모리에는 기본적으로 폴더는 구분되어 있지만 파일은 존재하지 않습니다. 테스트를 위해서 이미지 파일 몇 개를 에뮬레이터 메모리로 복사하겠습니다. 이클립스 왼쪽 아래 Connection Explorer에서 폴더 트리 목록을 확인할 수 있습니다. 이미지를 저장하는 공유폴더의 경로는 /opt/usr/media/Images입니다. 이 폴더를 선택하고 위쪽 툴바 중에서 'Push the file to the connected target device'를 선택합니다.

파일 선택 팝업창이 나타나면 부록 /Image 폴더로 이동해서 3개의 이미지 파일(0.jpg, 1.jpg, 2.jpg)을 선택하고 OK 버튼을 누릅니다. 본인이 사용하고 싶은 파일을 선택해도 상관없습니다.



팝업창이 닫히면 /Images 폴더에 파일이 복사됩니다.



2) 내장 메모리 폴더 목록 구하기

에뮬레이터 내장 메모리의 폴더 목록을 구해 보겠습니다. src 폴더에 소스파일(~.c)을 열고 define 상수와 변수를 추가합니다.

```
#include "filelist.h"

#define FM_PHONE_FOLDER    "/opt/usr/media"
#define FM_MEMORY_FOLDER   "/opt/storage/sdcard"

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    Evas_Object *list;
    char *current_path;
} appdata_s;
```

"/opt/usr/media" 는 내장 메모리의 공유폴더 루트 경로입니다.

"/opt/storage/sdcard" 는 외장 메모리의 공유폴더 루트 경로입니다.

list는 파일 목록을 표시하는 List 위젯 변수입니다.

current_path는 현재 폴더의 절대 경로를 저장하는 문자열 변수입니다.

create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

ad->current_path = calloc(PATH_MAX, sizeof(char));
strcpy(ad->current_path, FM_PHONE_FOLDER );
read_dir( ad );
}
```

current_path에 메모리를 할당하고 내장 메모리 루트 폴더 경로를 복사하였습니다.

read_dir() 는 특정 폴더 내부의 파일 목록을 구하는 함수입니다.
지금부터 만들어 보겠습니다.

create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```
static void
read_dir(appdata_s *ad)
{
    DIR *dir = opendir(ad->current_path);
    if( !dir )
        return;

    struct dirent *pDirent = NULL;
    char buf[100];
```

```
while ((pDirent = readdir(dir)) != NULL)
{
    if( pDirent->d_type == DT_DIR ) {
        dlog_print(DLOG_INFO, "tag", "[Folder] %s", pDirent->d_name);
    }
    else {
        dlog_print(DLOG_INFO, "tag", "[File] %s", pDirent->d_name);
    }
}
closedir(dir);
```

DIR은 폴더를 제어할 수 있는 구조체입니다. 파일 목록 읽기, 삭제, 폴더 생성 등이 가능합니다.

opendir (char *) 는 특정 폴더를 제어할 수 있는 DIR 객체를 반환하는 API입니다.

dirent는 파일(혹은 폴더) 정보를 저장하는 구조체입니다. 속성 중에서 d_name에는 파일명이 저장되어 있습니다. d_type에는 타입이 저장되어 있습니다. DT_DIR 이면 폴더이고, 그렇지 않으면 파일입니다.

readdir (DIR *) 는 특정 폴더 내부의 파일 목록을 읽어서 dirent 형식으로 1개씩 반환하는 API입니다. 파일 목록 끝까지 도달했다면 NULL을 반환합니다.

그 다음은 파일과 폴더를 구분해서 Log 메시지로 출력하는 코드입니다.

closedir (DIR *) 는 DIR을 닫는 API입니다.

예제를 빌드하고 실행시켜 보겠습니다. Log 패널에 내장 메모리 공용

폴더에 존재하는 파일과 폴더 목록이 나타납니다.

The screenshot shows the Eclipse IDE's Log view window titled "emulator-26101 (m-0604-1)". The window contains a table with columns: Time, Level, Pid, Tid, Tag, and Message. The data in the table is as follows:

Time	Level	Pid	Tid	Tag	Message
07-03 10:48:51.145	Info	2600	2600	tag	[Folder] Downloads
07-03 10:48:51.145	Info	2600	2600	tag	[Folder] Images
07-03 10:48:51.145	Info	2600	2600	tag	[Folder] Music
07-03 10:48:51.145	Info	2600	2600	tag	[Folder] Documents
07-03 10:48:51.155	Info	2600	2600	tag	[Folder] Videos
07-03 10:48:51.155	Info	2600	2600	tag	[Folder] ..
07-03 10:48:51.155	Info	2600	2600	tag	[Folder] .
07-03 10:48:51.155	Info	2600	2600	tag	[Folder] Sounds
07-03 10:48:51.155	Info	2600	2600	tag	[Folder] DCIM
07-03 10:48:51.155	Info	2600	2600	tag	[Folder] Others

Below the table, there is a search interface with a dropdown labeled "Tag" and a text input field containing "tag".

3) List 위젯에 파일 목록 추가

파일 목록을 List 위젯의 항목으로 추가해 보겠습니다. `create_base_gui()` 함수에 Box와 List 위젯 생성 코드를 입력합니다. Label 위젯은 주석 처리하겠습니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
```

```

/* Box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* List */
    ad->list = elm_list_add(ad->conform);
    elm_list_mode_set(ad->list, ELM_LIST_COMPRESS);
    evas_object_size_hint_weight_set(ad->list, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(ad->list, EVAS_HINT_FILL,
EVAS_HINT_FILL);
    elm_box_pack_end(box, ad->list);
    evas_object_show(ad->list);
}
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

read_dir() 함수를 다음과 같이 수정합니다.

```

static void
read_dir(appdata_s *ad)
{
    DIR *dir = opendir(ad->current_path);
    if( !dir )
        return;

    struct dirent *pDirent = NULL;

```

```

char buf[100];
elm_list_clear(ad->list);

while ((pDiren = readdir(dir)) != NULL)
{
    if( pDiren->d_type == DT_DIR ) {
        dlog_print(DLOG_INFO, "tag", "[Folder] %s", pDiren->d_name);
        sprintf(buf, "[ %s", pDiren->d_name);
    }
    else {
        dlog_print(DLOG_INFO, "tag", "[File] %s", pDiren->d_name);
        sprintf(buf, "# %s", pDiren->d_name);
    }
    elm_list_item_append(ad->list, buf, NULL, NULL, NULL, ad);
}
closedir(dir);
}

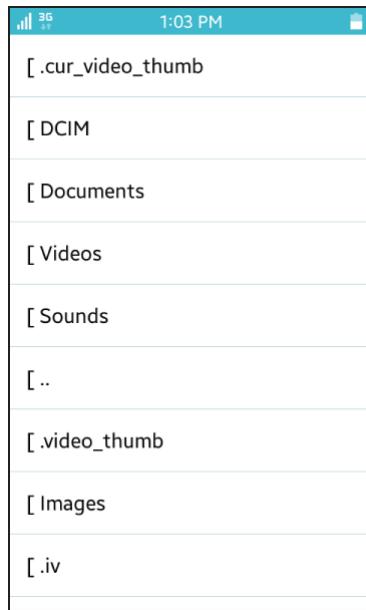
```

`elm_list_clear(Evas_Object *)` 는 List 위젯의 항목을 모두 삭제하는 API입니다.

폴더와 파일을 구분하기 위해서 컨텐츠가 폴더인 경우에는 이름 앞에 '[' 기호를 붙이고, 파일인 경우에는 이름 앞에 '#' 기호를 붙였습니다.

`elm_list_item_append(Evas_Object *, char *, Evas_Object *, Evas_Object *, Evas_Smart_Cb , void *)` 는 List 위젯에 새로운 항목을 추가하는 API입니다.

예제를 다시 실행시켜 봅시다. List 위젯에 루트 폴더 내부의 폴더와 파일 목록이 추가되었습니다. '.' 기호는 현재 폴더를 의미하고, '..' 기호는 상위 폴더를 의미합니다.



4) 폴더 이동

List 위젯의 항목을 선택하면 해당 폴더로 이동하는 기능을 구현해 보겠습니다. 그러기 위해서 다음 기능이 필요합니다.

- List 위젯 항목 선택 이벤트 콜백 함수
- 컨텐츠 목록에서 '.' 기호와 '..' 기호를 제거
- 현재 폴더가 루트 폴더가 아닌 경우에는 첫 번째 항목에 '..' 기호를 추가

read_dir() 함수에 새로운 코드를 추가합니다.

```
static void  
read_dir(appdata_s *ad)  
{  
    DIR *dir = opendir(ad->current_path);  
    if( !dir )
```

```

return;

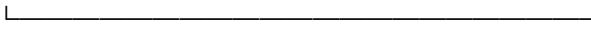
struct dirent *pDirent = NULL;
char buf[100];
elm_list_clear(ad->list);

if( strcmp(ad->current_path, FM_PHONE_FOLDER) != 0 )
    elm_list_item_append(ad->list, "..", NULL, NULL, list_item_clicked, ad);

while ((pDirent = readdir(dir)) != NULL)
{
    if( strcmp(pDirent->d_name, ".") == 0 )
        continue;
    if( strcmp(pDirent->d_name, "..") == 0 )
        continue;

    if( pDirent->d_type == DT_DIR ) {
        dlog_print(DLOG_INFO, "tag", "[Folder] %s", pDirent->d_name);
        sprintf(buf, "[ %s", pDirent->d_name);
    }
    else {
        dlog_print(DLOG_INFO, "tag", "[File] %s", pDirent->d_name);
        sprintf(buf, "# %s", pDirent->d_name);
    }
    elm_list_item_append(ad->list, buf, NULL, NULL, list_item_clicked, ad);
    //elm_list_item_append(ad->list, buf, NULL, NULL, NULL, ad);
}
closedir(dir);
}

```



현재 폴더가 내장 메모리 루트 폴더인 경우에 '..' 기호를 List 위젯 첫 번째 항목으로 추가합니다.

컨텐츠 이름이 '.' 기호와 '..' 기호인 경우에는 무시합니다.

항목 선택 이벤트 콜백 함수명을 `list_item_clicked` 으로 지정합니다.
지금부터 이 함수를 만들어 보겠습니다. `read_dir()` 함수 위에 3개의
함수를 생성합니다.

```
static char*
get_file_name(const char* item_text, bool *is_file)
{
    if( item_text[0] == '#' )
        *is_file = true;
    else
        *is_file = false;

    if( strcmp(item_text, "..") == 0 )
        return item_text;
    return item_text + 2;
}

static char*
get_new_path(char *current_path, const char *folder_name)
{
    if( strcmp(folder_name, "..") == 0 )
    {
        int pos = strlen( current_path ) - strlen( strrchr( current_path, '/' ) );
        current_path[pos] = '\0';
    }
    else
        sprintf(current_path, "%s/%s", current_path, folder_name);
    return current_path;
}

static void
```

```

list_item_clicked(void *data, Evas_Object *obj, void *event_info)
{
    Elm_Object_Item *it = event_info;
    const char *item_text = elm_object_item_text_get(it);

    bool is_file;
    char *file_name = get_file_name(item_text, &is_file);
    if( is_file )
        return;

    appdata_s *ad = data;
    ad->current_path = get_new_path(ad->current_path, file_name);

    read_dir( ad );
}

```

get_file_name() 은 List 항목의 텍스트를 받아서 기호를 제거하고 폴더인지 파일인지를 구분하는 함수입니다.

첫글자가 '#' 이면 파일이고, '[' 이면 폴더입니다. 텍스트가 '..'이면 상위 폴더를 의미합니다.

get_new_path() 는 현재 경로와 새로운 폴더 명을 받아서 전체 폴더 경로를 구하는 함수입니다.

strrchr (char *, int) 는 문자열에 특정 문자를 검색하는 API입니다. 뒤쪽부터 검색을 시작합니다. 그래서 가장 뒤쪽에 있는 문자의 포인터를 반환합니다.

새로운 폴더명이 '..' 이면 상위 폴더를 의미하기 때문에 현재 폴더 경로에서 마지막 폴더 이름을 삭제합니다. 그렇지 않다면 현재 폴더 경로 뒤에 새로운 폴더명을 추가해서 절대 경로를 생성합니다.

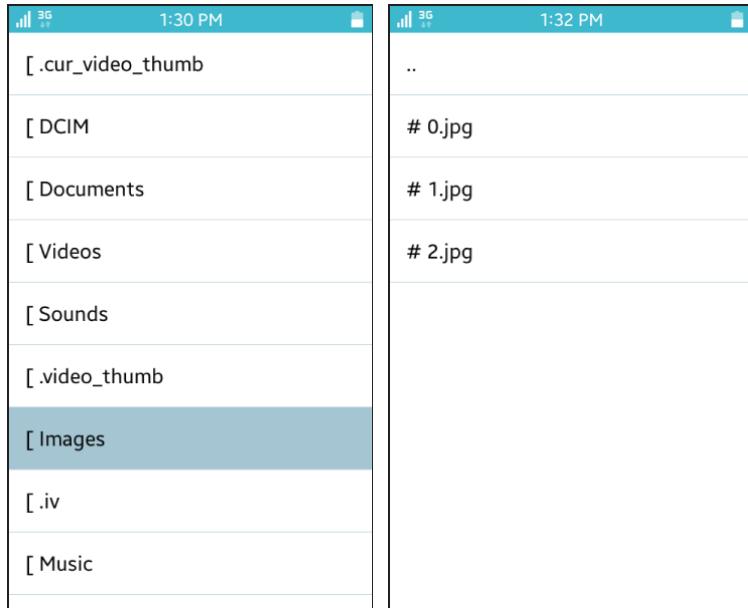
`list_item_clicked()` 는 List 위젯 항목 선택 이벤트 함수입니다. 해당 항목의 캡션 텍스트를 구해서 컨텐츠 이름을 분리한 다음에, 새로운 폴더 경로를 만들어서 그 폴더 안에 컨텐츠를 List 위젯에 추가합니다.

이 상태로 빌드하면 오류가 발생합니다. 이유는 `read_dir()` 함수와 `list_item_clicked()` 함수가 서로를 호출하기 때문입니다. 이런 경우에는 함수의 헤더를 선언해주면 됩니다. 위치는 소스파일 제일 위쪽 또는 헤더파일입니다. 이번 예제에서는 소스파일 위쪽에 선언하겠습니다. 소스파일 위쪽으로 이동해서 `read_dir()` 함수를 선언해 줍니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    Evas_Object *list;
    char *current_path;
} appdata_s;

static void read_dir(appdata_s *ad);
```

예제를 다시 실행하면 폴더 목록에서 '.' 와 '..' 기호가 사라졌습니다. Images 항목을 선택하면 좀전에 복사했던 파일 목록이 나타납니다. 1번째 항목에는 '..'가 추가되었습니다. 이것을 선택하면 이전 폴더로 되돌아 갑니다.



5) ELM File Selector

ELM File Selector 위젯을 사용하면 간편하게 파일 관리자를 구현할 수 있습니다. 새로운 소스 프로젝트를 생성하고 이름을 ElmFileSelectorEx 이라고 지정합니다.

소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 다음 Privilege를 추가합니다.

<http://tizen.org/privilege/mediastorage>

소스파일(/src/elmfileselectorex.c)을 열고 create_base_gui() 함수에 새로운 코드를 추가합니다. Box와 FileSelector 위젯을 생성하는 코드입니다. Label 위젯 생성 코드는 삭제합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* Box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

Evas_Object *fs = elm_fileselector_add(ad->conform);
evas_object_size_hint_weight_set(fs, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(fs, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_box_pack_end(box, fs);
evas_object_show(fs);

elm_fileselector_path_set(fs, FM_PHONE_FOLDER);
//elm_fileselector_expandable_set(fs, EINA_TRUE);
elm_fileselector_is_save_set(fs, EINA_FALSE);
elm_fileselector_mode_set(fs, ELM_FILESELECTOR_LIST);
elm_fileselector_folder_only_set(fs, EINA_FALSE);
}

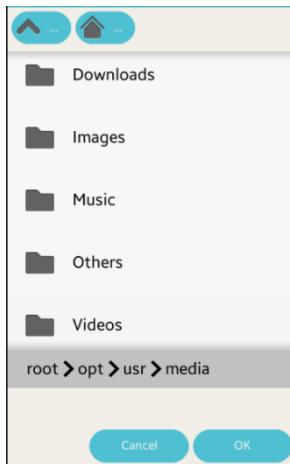
}

/* Show window after base gui is set up */

```

```
    evas_object_show(ad->win);  
}  
_____
```

예제를 빌드하고 실행시켜 봅시다. FileSelector 위젯에 폴더 목록이 나타납니다.



FileChooser 위젯에 대한 자세한 사용방법은 아래 링크를 참조하기 바랍니다.

https://docs.enlightenment.org/elementary/1.15.0/fileselector_example.html

https://docs.enlightenment.org/elementary/1.15.0/group_Fileselector.html

6) 관련 API

DIR : 폴더를 제어할 수 있는 구조체. 파일 목록 읽기, 삭제, 폴더 생성 등이 가능합니다.

`DIR *opendir (char *_name)` : 특정 폴더를 제어할 수 있는 DIR 객체를 반환하는 API.

`dirent` : 파일(혹은 폴더) 정보를 저장하는 구조체. 속성 중에서 `d_name`에는 파일명이 저장되어 있습니다. `d_type`에는 타입이 저장되어 있습니다. `DT_DIR`이면 폴더이고, 그렇지 않으면 파일입니다.

`struct dirent *readdir (DIR *_dirp)` : 특정 폴더 내부의 파일 목록을 읽어서 `dirent` 형식으로 1개씩 반환하는 API. 파일 목록 끝까지 도달했다면 `NULL`을 반환합니다.

`int closedir (DIR *_dirp)` : DIR을 닫는 API.

`void elm_list_clear(Evas_Object *obj)` : List 위젯의 항목을 모두 삭제하는 API.

`Elm_Object_Item *elm_list_item_append(Evas_Object *obj, const char *label, Evas_Object *icon, Evas_Object *end, Evas_Smart_Cb func, const void *data)` : List 위젯에 새로운 항목을 추가하는 API.

`char *strrchr (char *_s, int _c)` : 문자열에 특정 문자를 검색하는 API. 뒤쪽부터 검색을 시작합니다. 그래서 가장 뒤쪽에 있는 문자의 포인터를 반환합니다.

60. Preference 사용방법

앱의 환경설정 같은 정보는 레지스터리에 저장해 놓으면 편리합니다. Preference는 로컬에 데이터를 저장할 수 있는 저장공간입니다. 환경설정 정보를 저장하기에 적당한 용도입니다. 앱이 삭제되면 해당 Preference도 함께 삭제됩니다.

1) Preference에 문자열 데이터 저장

새로운 소스 프로젝트를 생성하고 Project name을 PreferenceEx으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "preferenceex.h"
#include <app_preference.h>
#include <stdlib.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *entry1;
    Evas_Object *spinner1;
} appdata_s;

const char *string_key = "string_key";
const char *integer_key = "integer_key";
```

app_preference.h는 Preference를 사용하기 위한 라이브러리

헤더파일입니다.

stdlib.h는 문자열과 숫자의 형변환을 위한 라이브러리 헤더파일입니다.

entry1에는 문자열 데이터를 입력하고, spinner1에는 숫자 데이터를 입력하겠습니다.

화면에 몇가지 위젯을 추가하겠습니다. create_base_gui() 함수 위에 2개의 새로운 함수 생성합니다.

```
static void  
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)  
{  
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);  
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);  
    elm_table_pack(table, child, x, y, w, h);  
    evas_object_show(child);  
}  
  
static Evas_Object *  
my_button_add(Evas_Object *parent, const char *text, Evas_Smart_Cb cb, void *cb_data)  
{  
    Evas_Object *btn;  
  
    btn = elm_button_add(parent);  
    elm_object_text_set(btn, text);  
    evas_object_smart_callback_add(btn, "clicked", cb, cb_data);  
  
    return btn;  
}
```

my_table_pack() 은 Table에 위젯을 추가하는 함수입니다.

my_button_add() 은 Button 위젯을 생성하는 함수입니다.

그런 다음 create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

    /* Box to put the table in so we can bottom-align the table
     * window will stretch all resize object content to win size */
    Evas_Object *box = elm_box_add(ad->conform);
    evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND, 0.0);
    elm_object_content_set(ad->conform, box);
    evas_object_show(box);

    /* Table */
    Evas_Object *table = elm_table_add(ad->conform);
    /* Make table homogenous - every cell will be the same size */
    elm_table_homogeneous_set(table, EINA_TRUE);
    /* Set padding of 10 pixels multiplied by scale factor of UI */
    elm_table_padding_set(table, 10 * elm_config_scale_get(), 30 *
elm_config_scale_get());
    /* Let the table child allocation area expand within in the box */
    evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND, 0.0);
    /* Set table to fill width but align to bottom of box */
    evas_object_size_hint_align_set(table, EVAS_HINT_FILL, 0.0);
    elm_box_pack_end(box, table);
    evas_object_show(table);
```

```

{

/* Label-1 */
Evas_Object *label = elm_label_add(ad->conform);
elm_object_text_set(label, "Pet name:");
my_table_pack(table, label, 0, 0, 1, 1);

/* Bg-1 */
Evas_Object *bg = elm_bg_add(ad->conform);
elm_bg_color_set(bg, 210, 210, 210);
my_table_pack(table, bg, 1, 0, 1, 1);

/* Entry-1 */
ad->entry1 = elm_entry_add(ad->conform);
my_table_pack(table, ad->entry1, 1, 0, 1, 1);

/* Label-2 */
label = elm_label_add(ad->conform);
elm_object_text_set(label, "Percentage:");
my_table_pack(table, label, 0, 1, 1, 1);

/* Spinner-1 */
ad->spinner1 = elm_spinner_add(ad->conform);
elm_spinner_editable_set(ad->spinner1, EINA_TRUE);
elm_spinner_interval_set(ad->spinner1, 1);
elm_spinner_min_max_set(ad->spinner1, 0, 100);
elm_spinner_label_format_set(ad->spinner1, "%.0f");
my_table_pack(table, ad->spinner1, 1, 1, 1, 1);

Evas_Object *btn;

/* Button-Save */
btn = my_button_add(ad->conform, "Save", btn_save_cb, ad);
my_table_pack(table, btn, 0, 3, 2, 1);

}

```

```
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

}
```

Box, Table, 2개의 Label, Bg, Entry, Spinner 그리고 1개의 Button 위젯을 추가하였습니다. 1번째 Entry에 문자열을 입력하고 Button을 누르면 사용자가 입력한 문자열을 Preference에 저장하겠습니다.

create_base_gui() 함수 위에 Button 콜백 함수를 생성합니다.

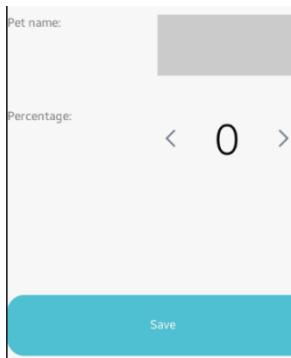
```
static void
btn_save_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    const char *string_value;
    int integer_value;

    string_value = elm_object_text_get(ad->entry1);
    preference_set_string(string_key, string_value);
}
```

preference_set_string(char *, char *) 는 Preference에 문자열 데이터를 저장하는 API입니다. 1번째 파라미터에는 Key를 전달하고, 2번째 파라미터에는 문자열 데이터를 전달합니다. Key는 Read와 Write를 할 때 동일해야 합니다.

예제를 빌드하고 실행시켜 봅시다. 1번째 Entry에 문자열을 입력하고 Save Button을 눌러봅시다. 입력한 문자열이 저장되었습니다. 아쉽게도

읽는 기능은 아직 구현되지 않았습니다. 지금부터 만들어 보겠습니다.



2) 문자열 Preference 읽기

2번째 Button을 추가해서 Preference에서 데이터를 읽는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다.

```
    Evas_Object *btn;  
  
    /* Button-Load */  
    btn = my_button_add(ad->conform, "Load", btn_read_cb, ad);  
    my_table_pack(table, btn, 0, 2, 2, 1);  
  
    /* Button-Save */  
    btn = my_button_add(ad->conform, "Save", btn_save_cb, ad);  
    my_table_pack(table, btn, 0, 3, 2, 1);  
}  
}
```

새로 추가된 Button의 콜백 함수를 만들어 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void
btn_read_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char *string_value = "";
    bool existing = false;

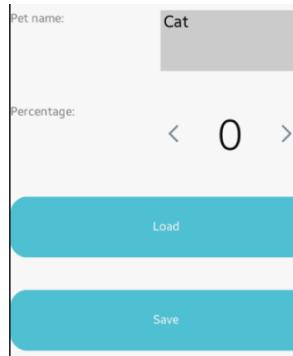
    if ((preference_is_existing(string_key, &existing) == 0) && existing)
    {
        preference_get_string(string_key, &string_value);
        elm_object_text_set(ad->entry1, string_value);
        free(string_value);
    }
}
```

Preference에서 데이터를 읽을 때는 먼저 해당 데이터가 존재하는지를 확인해야 합니다.

`preference_is_existing(const char *, bool *)` 는 Preference에 특정 데이터가 존재하는지 확인하는 API입니다. 1번째 파라미터에 Key를 전달하면 2번째 파라미터에선 데이터 존재 여부를 반환합니다.

`preference_get_string(char *, char **)` 는 Preference에서 문자열 데이터를 읽는 API입니다. 1번째 파라미터에 Key를 전달하면 2번째 파라미터에서 문자열 데이터를 반환합니다.

예제를 다시 실행하고 1번째 Entry에 문자열을 입력하고 Save 버튼을 누릅니다. 그런 다음 Entry에 입력된 문자를 삭제하고 Read 버튼을 누릅니다. 좀전에 입력했던 문자열이 Entry에 표시됩니다.



3) Preference 숫자 Read & Write

이번에는 Preference에 숫자를 저장했다가 다시 읽어보겠습니다. 먼저 저장 함수에 다음과 같이 새로운 코드를 추가합니다.

```
static void
btn_save_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    const char *string_value;
    int integer_value;

    string_value = elm_object_text_get(ad->entry1);
    preference_set_string(string_key, string_value);

    integer_value = (int) elm_spinner_value_get(ad->spinner1);
    preference_set_int(integer_key, integer_value);
}
```

atoi (char *) 는 Array to Int의 약자로서 문자열을 숫자로 변경하는 API입니다.

`preference_set_int(char *, int)` 는 Preference에 정수형 데이터를 저장하는 API입니다.

데이터를 읽는 함수도 새로운 코드를 추가합니다.

```
static void
btn_read_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char *string_value = "";
int integer_value;
    bool existing = false;

    if ((preference_is_existing(string_key, &existing) == 0) && existing)
    {
        preference_get_string(string_key, &string_value);
        elm_object_text_set(ad->entry1, string_value);
        free(string_value);
    }

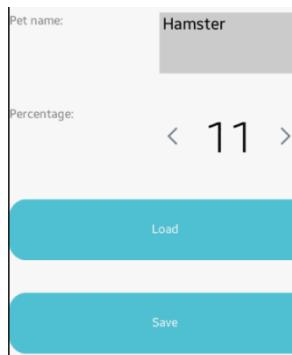
if ((preference_is_existing(integer_key, &existing) == 0) && existing)
{
    preference_get_int(integer_key, &integer_value);
    elm_spinner_value_set(ad->spinner1, (double) integer_value);
}
}
```

`preference_get_int(char *, int *)` 는 Preference에서 정수형 데이터를 읽는 API입니다.

`eina_convert_itoa(int, char *)` 는 정수를 문자열로 변환하는 API입니다.

예제를 다시 실행시켜 봅시다. 1번째 Entry에 문자열을 입력하고 2번째 Entry에 숫자를 입력한 다음, Save 버튼을 누릅니다.

입력 데이터를 변경했다가 다시 실행하고 Read 버튼을 누르면 Preference에 저장되었던 데이터가 다시 표시됩니다.



4) 관련 API

`int preference_set_string(const char *key, const char *value)` : Preference에 문자열 데이터를 저장하는 API. 1번째 파라미터에는 Key를 전달하고, 2번째 파라미터에는 문자열 데이터를 전달합니다. Key는 Read와 Write를 할 때 동일해야 합니다.

`int preference_is_existing(const char *key, bool *existing)` : Preference에 특정 데이터가 존재하는지 확인하는 API. 1번째 파라미터에 Key를 전달하면 2번째 파라미터에선 데이터 존재 여부를 반환합니다.

`int preference_get_string(const char *key, char **value)` : Preference에서 문자열 데이터를 읽는 API. 1번째 파라미터에 Key를 전달하면 2번째 파라미터에서 문자열 데이터를 반환합니다.

int atoi (char *_nptr) : Array to Int의 약자로서 문자열을 숫자로 변경하는 API.

int preference_set_int(char *key, int value) : Preference에 정수형 데이터를 저장하는 API.

int preference_get_int(char *key, int *value) : Preference에서 정수형 데이터를 읽는 API.

int eina_convert_itoa(int n, char *s) : 정수를 문자열로 변환하는 API.

61. SQLite로 만드는 성적표 예제

사용자 환경설정 정보는 Preference를 사용해서 저장해도 되지만, 일정 관리나 연락처, 성적 관리 처럼 체계적이고 방대한 데이터는 데이터베이스를 사용해야 합니다. 대부분의 모바일 플랫폼에서는 자체적으로 SQLite를 DB로 제공하고 있으며 타이젠에서도 SQLite가 지원됩니다. SQLite는 오라클이나 MS-SQL에 비해서 최대 데이터 용량이 적은 편이지만 모바일 기기에서는 이 정도만으로도 충분합니다. 사용방법이 간편하고 표준 SQL 쿼리 구문을 사용하기 때문에 다른 시스템에서 사용하던 DB를 옮겨왔을 때 호환성이 좋습니다.

하나의 데이터베이스는 1개 이상의 테이블을 포함하고 있습니다. 테이블은 엑셀시트와 유사한 2차원 표 라고 생각하면 됩니다. 가로 방향으로 여러 개의 컬럼이 늘어서 있고, 개별 필드의 집합으로 구성된 데이터를 채울 때마다 세로 방향으로 레코드가 한줄씩 늘어나는 것입니다. 이번 시간에는 SQLite를 사용해서 간단한 성적표 예제를 만들어 보겠습니다.

1) DB 생성

새로운 소스 프로젝트를 생성하고 Project name을 SqliteEx 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수, 그리고 구조체를 추가합니다.

```
#include "sqliteex.h"  
#include <sqlite3.h>  
#include <stdlib.h>
```

```
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    Evas_Object *entry1;  
    Evas_Object *entry2;  
    Evas_Object *entry3;  
    Evas_Object *list;  
  
    sqlite3 *db; // Database handle  
    char *current_key;  
} appdata_s;
```

```
typedef struct recdata {  
    char key[10];  
    char name[255];  
    char english[10];  
    char math[10];  
} recdata_s;
```

```
appdata_s *m_ad;
```

sqlite3.h는 SQLite를 사용하기 위한 라이브러리 헤더파일입니다.

stdlib.h는 문자열과 숫자를 형변환 하기 위한 라이브러리 헤더파일입니다.

appdata 구조체에 3개의 Entry를 추가하였습니다. 각각 이름, 영어 점수, 수학 점수를 입력하겠습니다.

list 위젯에는 성적표 DB 목록을 출력하겠습니다.

sqlite3는 DB 객체 변수입니다.

current_key에는 현재 선택된 레코드의 Key 값을 저장하겠습니다.

reodata는 학생의 성적 데이터를 저장하기 위한 구조체입니다. reodata 1개가 하나의 레코드에 해당됩니다.

m_ad는 appdata를 어디서든 접근하기 위해서 전역변수로 선언하였습니다.

/data 폴더에 DB 파일을 생성하고 성적표 테이블을 생성하겠습니다.
create_base_gui() 함수 위에 새로운 함수 3개를 생성합니다.

```
static int CreateTable(appdata_s *ad)
{
    char *ErrMsg;
    char *sql = "CREATE TABLE IF NOT EXISTS ReportCard(KEY INTEGER PRIMARY KEY,
NAME TEXT NOT NULL, ENGLISH INT NOT NULL, MATH INT NOT NULL);";

    int ret = sqlite3_exec(ad->db, sql, NULL, 0, &ErrMsg);
    return ret;
}

static void
init_db(appdata_s *ad)
{
    sqlite3_shutdown();
    sqlite3_config(SQLITE_CONFIG_URI, 1);
    sqlite3_initialize();
    char * resource = app_get_data_path();
    int siz = strlen(resource) + 10;
    char * path = malloc(sizeof(char)*siz);
    strncat(path, resource, siz);
```

```

strncat(path, "test.db", siz);

sqlite3_open(path, &ad->db);
free(path);

CreateTable(ad);

}

static void
my_table_pack(Evas_Object *table, Evas_Object *child, int x, int y, int w, int h)
{
    evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
    evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND);
    elm_table_pack(table, child, x, y, w, h);
    evas_object_show(child);
}

```

CreateTable() 는 SQL 쿼리 구문을 사용해서 테이블을 생성하는 함수입니다.

쿼리 구문에서 'CREATE TABLE' 은 테이블을 생성하는 명령어입니다.

'IF NOT EXISTS' 는 테이블이 존재하지 않을 때에 생성하라는 의미입니다.

테이블 이름은 ReportCard 으로 지정하였습니다.

'KEY INTEGER PRIMARY KEY' 는 Key 컬럼을 추가하는 코드입니다. 형식은 숫자이고, 레코드가 추가될 때마다 숫자가 자동으로 증가합니다.

'NAME TEXT NOT NULL' 는 이름을 저장하는 컬럼을 추가하는 코드입니다. 형식은 텍스트이고, 비워둘 수 없습니다.

'ENGLISH INT NOT NULL' 는 영어 점수를 저장하는 컬럼을 추가하는 코드입니다. 형식은 숫자이고, 비워둘 수 없습니다.

'MATH INT NOT NULL' 는 수학 점수를 저장하는 컬럼을 추가하는 코드입니다. 형식은 숫자이고, 비워둘 수 없습니다.

`sqlite3_exec(sqlite3*, char *, int (*callback), void *, char **)` 는 SQL 쿼리문을 실행하는 API입니다. 파라미터는 순서대로 SQLite 객체, 쿼리문, 콜백 함수명, 사용자 데이터, 에러메시지 반환 입니다.

`init_db()` 는 DB 파일을 생성하는 함수입니다.

`sqlite3_shutdown()` 는 DB를 닫는 API입니다.

`sqlite3_config(int, ...)` 는 DB 속성을 지정하는 API입니다.

`SQLITE_CONFIG_URI`를 전달하면 DB 파일에 데이터를 저장할 수 있습니다.

`sqlite3_initialize()` DB를 초기화하는 API 입니다.

`app_get_data_path()` 는 /data 폴더의 절대 경로를 반환하는 API입니다. /res 폴더에는 Write가 허용되지 않기 때문에 /data 폴더에 DB 파일을 생성하겠습니다.

`strncat (char *, char *, size_t)` 는 최대 길이를 지정해서 문자열 뒤에 새로운 문자열을 추가하는 API입니다. 파라미터는 순서대로 원본 문자열 배열, 추가될 문자열 데이터, 최대 길이 입니다.

`sqlite3_open(char *, sqlite3 **)` 는 DB 파일을 Open 하는 API입니다. DB 파일이 존재하지 않으면 생성합니다. 1번째 파라미터에 파일 경로를 전달하면, 2번째 파라미터에서 DB 객체를 반환합니다.

`my_table_pack()` 은 Table에 위젯을 추가하는 함수입니다.

앱이 실행되면 자동으로 DB 파일을 Open 하겠습니다. `create_base_gui()` 함수 끝부분에서 위 함수를 호출하면 됩니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

init_db(ad);

}
```

2) 새로운 레코드 추가

3개의 Entry 위젯에 이름, 영어 성적, 수학 성적을 입력하고 Button을 누르면 새로운 레코드가 DB에 추가되는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 새로운 코드를 추가합니다. Label 위젯은 주석 처리합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* Box to put the table in so we can bottom-align the table
```

```

 * window will stretch all resize object content to win size */
Evas_Object *box = elm_box_add(ad->conform);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND, 0.0);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

/* Table */
Evas_Object *table = elm_table_add(ad->conform);
/* Make table homogenous - every cell will be the same size */
elm_table_homogeneous_set(table, EINA_TRUE);
/* Set padding of 10 pixels multiplied by scale factor of UI */
elm_table_padding_set(table, 10 * elm_config_scale_get(), 10 *
elm_config_scale_get());
/* Let the table child allocation area expand within in the box */
evas_object_size_hint_weight_set(table, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
/* Set table to fiill width but align to bottom of box */
evas_object_size_hint_align_set(table, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_box_pack_end(box, table);
evas_object_show(table);

{
    /* Bg-1 */
    Evas_Object *bg = elm_bg_add(ad->conform);
    elm_bg_color_set(bg, 210, 210, 210);
    my_table_pack(table, bg, 0, 0, 1, 1);

    /* Entry-1 */
    ad->entry1 = elm_entry_add(ad->conform);
    elm_object_part_text_set(ad->entry1, "elm.guide", "Name");
    my_table_pack(table, ad->entry1, 0, 0, 1, 1);

    /* Bg-2 */
    bg = elm_bg_add(ad->conform);
    elm_bg_color_set(bg, 210, 210, 210);

```

```

my_table_pack(table, bg, 1, 0, 1, 1);

/* Entry-2 */
ad->entry2 = elm_entry_add(ad->conform);
elm_object_part_text_set(ad->entry2, "elm.guide", "English");
my_table_pack(table, ad->entry2, 1, 0, 1, 1);

/* Bg-3 */
bg = elm_bg_add(ad->conform);
elm_bg_color_set(bg, 210, 210, 210);
my_table_pack(table, bg, 2, 0, 1, 1);

/* Entry-3 */
ad->entry3 = elm_entry_add(ad->conform);
elm_object_part_text_set(ad->entry3, "elm.guide", "Math");
my_table_pack(table, ad->entry3, 2, 0, 1, 1);

/* Button-Add */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Add");
evas_object_smart_callback_add(btn, "clicked", btn_add_cb, ad);
my_table_pack(table, btn, 0, 1, 1, 1);

}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

Box, Table, 3개의 Bg와 3개의 Entry 위젯을 생성하였습니다. 그리고 1개의 Button 위젯을 생성하였습니다.

Button을 누르면 Entry에 입력된 데이터를 DB에 추가하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수 2개를

생성합니다.

```
static int  
InsertRecord(appdata_s *ad, unsigned char *name, int english, int math)  
{  
    char sql[256];  
    char *ErrMsg;  
    snprintf(sql, 256, "INSERT INTO ReportCard VALUES(NULL, %s, %d, %d);", name,  
    english, math);  
    int ret = sqlite3_exec(ad->db, sql, NULL, 0, &ErrMsg);  
    return ret;  
}  
  
static void  
btn_add_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
  
    char* s_name = elm_object_text_get(ad->entry1);  
    char* s_english = elm_object_text_get(ad->entry2);  
    int n_english = atoi(s_english);  
    char* s_math = elm_object_text_get(ad->entry3);  
    int n_math = atoi(s_math);  
  
    InsertRecord(ad, s_name, n_english, n_math);  
}
```

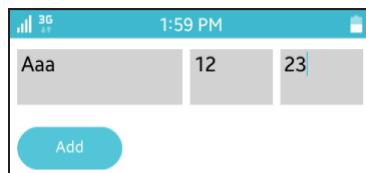
InsertRecord() 는 SQL 쿼리 구문을 사용해서 새로운 레코드를 추가하는 함수입니다.

쿼리 구문에서 'INSERT INTO ReportCard' 은 ReportCard 라는 이름의 테이블에 새로운 레코드를 추가하는 명령어입니다.

VALUES()에는 레코드 데이터를 전달합니다. 1번째 컬럼 Key 는 자동으로 생성되기 때문에 생략합니다.

btn_add_cb() 는 사용자가 Button을 눌렀을 때 Entry에 입력된 데이터를 구해서 DB에 저장하는 함수입니다.

예제를 빌드하고 실행시켜 보겠습니다. 1번째 Entry에는 이름을, 2번째 Entry에는 영어 점수, 3번째 Entry에는 수학 점수를 입력하고 Add 버튼을 누릅니다. DB에 새로운 레코드가 추가되었습니다. 안타깝게도 확인해 볼 수는 없습니다. DB에 저장된 데이터를 읽는 기능이 아직 구현되지 않았기 때문입니다.



3) DB 데이터 읽기

DB에 저장된 데이터를 읽어서 화면에 출력하는 기능을 구현해 보겠습니다. 그러기 위해서 create_base_gui() 함수에 List 위젯 생성 코드를 추가합니다.

```
static void  
create_base_gui(appdata_s *ad)  
{  
    m_ad = ad;
```

~

```

/* Button-Add */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Add");
evas_object_smart_callback_add(btn, "clicked", btn_add_cb, ad);
my_table_pack(table, btn, 0, 1, 1, 1);

/* List */
ad->list = elm_list_add(ad->conform);
elm_list_mode_set(ad->list, ELM_LIST_COMPRESS);
elm_list_go(ad->list);
my_table_pack(table, ad->list, 0, 2, 3, 8);

}

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

appdata 객체를 어디에서나 사용하기 위해서 전역변수에 저장하였습니다. 새로운 함수 2개를 추가합니다. 이 함수들은 init_db() 와 btn_add_cb() 보다 위쪽에 위치해야 합니다.

```

static int db_read_cb(void *counter, int argc, char **argv, char **azColName)
{
    char buf[255];

    recdata_s* rd = malloc(sizeof(recdata_s));
    strcpy(rd->key, argv[0]);
    strcpy(rd->name, argv[1]);
    strcpy(rd->english, argv[2]);
    strcpy(rd->math, argv[3]);

    sprintf(buf, "%s / %s / %s / %s", argv[0], argv[1], argv[2], argv[3]);

```

```
elm_list_item_append(m_ad->list, buf, NULL, NULL, NULL, (void*)rd);
elm_list_go(m_ad->list);
return 0;
}

static int read_db(appdata_s *ad)
{
    char *sql = "select * from ReportCard";
    int counter=0;
    char *ErrMsg;

    elm_list_clear(ad->list);
    int ret = sqlite3_exec(ad->db, sql, db_read_cb, &counter, &ErrMsg);
    return ret;
}
```

DB에서 여러개의 레코드를 읽을 때 개별 레코드 정보가 콜백 함수에 전달됩니다. db_read_cb() 는 1개의 레코드를 받아서 처리하는 콜백 함수입니다. 3번째 파라미터에 배열로 데이터가 전달됩니다.

reodata_s 구조체를 생성해서 각 필드 데이터를 저장하고 모든 데이터를 하나의 문자열로 합쳐서 List 위젯에 새로운 항목을 추가합니다.

read_db() 는 DB에 저장된 모든 데이터를 읽어서 List위젯에 출력하는 함수입니다.

'select * from ReportCard' 쿼리문은 ReportCard 테이블에 저장된 모든 데이터를 반환하라는 의미입니다.

위 함수를 앱이 실행될 때와 Button을 눌렀을 때 호출하면 됩니다. init_db() 와 btn_add_cb() 함수 끝부분에 새로운 코드를 추가합니다.

```

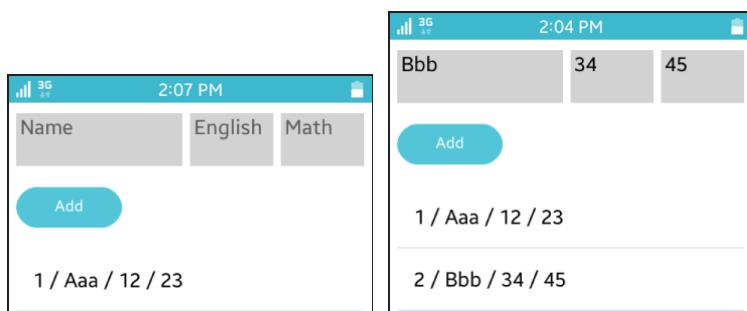
static void
init_db(appdata_s *ad)
{
    ~
    CreateTable(ad);
read_db(ad);
}

static void
btn_add_cb(void *data, Evas_Object *obj, void *event_info)
{
    ~
    InsertRecord(ad, s_name, n_english, n_math);
read_db(ad);
}

```

그렇기 때문에 `read_db()` 함수는 `init_db()` 와 `btn_add_cb()` 보다 위쪽에 위치해야 하는 것입니다.

예제를 다시 실행시켜 봅시다. 좀전에 입력했던 데이터가 List 위젯에 추가되었습니다. 새로운 데이터를 입력하고 Add 버튼을 눌러봅시다. List 위젯에 새 항목이 추가됩니다.



4) 데이터 수정

저장되어 있는 데이터를 수정하는 방법을 알아보겠습니다. 사용자가 List 항목을 선택하면 그 항목의 데이터를 Entry에 출력하는 기능을 구현하겠습니다. db_read_cb() 함수에서 새 항목을 추가하는 코드에 콜백 함수명을 지정합니다. 그런 다음 콜백 함수를 생성합니다.

```
static void
list_item_clicked(void *data, Evas_Object *obj, void *event_info)
{
    recdata_s* rd = (recdata_s*)data;
    m_ad->current_key = rd->key;
    elm_object_text_set(m_ad->entry1, rd->name);
    elm_object_text_set(m_ad->entry2, rd->english);
    elm_object_text_set(m_ad->entry3, rd->math);
}

static int db_read_cb(void *counter, int argc, char **argv, char **azColName)
{
    char buf[255];

    recdata_s* rd = malloc(sizeof(recdata_s));
    strcpy(rd->key, argv[0]);
    strcpy(rd->name, argv[1]);
    strcpy(rd->english, argv[2]);
    strcpy(rd->math, argv[3]);

    sprintf(buf, "%s / %s / %s / %s", argv[0], argv[1], argv[2], argv[3]);
    elm_list_item_append(m_ad->list, buf, NULL, NULL, list_item_clicked, (void*)rd);
    //elm_list_item_append(m_ad->list, buf, NULL, NULL, NULL, (void*)rd);
    elm_list_go(m_ad->list);
    return 0;
}
```

```
}
```

list_item_clicked() 는 List 위젯 항목 선택 이벤트 콜백 함수입니다. Key 값을 전역변수에 저장하고, 그외 데이터를 Entry 위젯에 출력합니다.

db_read_cb() 함수에서 List 위젯 항목 추가 코드를 수정하였습니다. 항목 선택 콜백 함수명을 지정하였습니다.

그런 다음 create_base_gui() 함수에 새로운 Button 생성 코드를 추가합니다.

```
/* Button-Add */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Add");
evas_object_smart_callback_add(btn, "clicked", btn_add_cb, ad);
my_table_pack(table, btn, 0, 1, 1, 1);

/* Button-Update */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Update");
evas_object_smart_callback_add(btn, "clicked", btn_update_cb, ad);
my_table_pack(table, btn, 1, 1, 1, 1);

/* List */
ad->list = elm_list_add(ad->conform);
elm_list_mode_set(ad->list, ELM_LIST_COMPRESS);
elm_list_go(ad->list);
my_table_pack(table, ad->list, 0, 2, 3, 8);
```

Button을 누르면 현재 선택된 항목의 데이터를 수정하는 기능을 구현하겠습니다. create_base_gui() 함수 위에 2개의 새로운 함수를

생성합니다.

```
static int
UpdateRecord(appdata_s *ad, unsigned char *name, unsigned char *english, unsigned
char *math)
{
    char sql[256];
    char *ErrMsg;
    snprintf(sql, 256, "UPDATE ReportCard SET NAME=%s", ENGLISH=%s",
MATH=%s WHERE KEY=%s;",
            name, english, math, ad->current_key);
    int ret = sqlite3_exec(ad->db, sql, NULL, 0, &ErrMsg);
    return ret;
}

static void
btn_update_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    char* s_name = elm_object_text_get(ad->entry1);
    char* s_english = elm_object_text_get(ad->entry2);
    char* s_math = elm_object_text_get(ad->entry3);

    UpdateRecord(ad, s_name, s_english, s_math);

    read_db(ad);
}
```

UpdateRecord() 는 DB에 저장된 데이터를 수정하는 함수입니다.

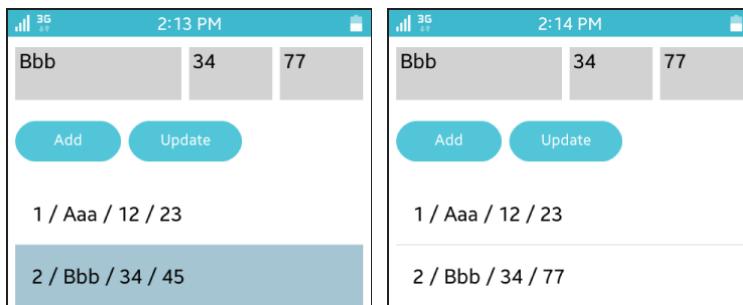
쿼리문에서 'UPDATE ReportCard' 는 ReportCard 테이블에 저장된 데이터를 수정한다는 의미입니다.

SET NAME='%'s' 는 NAME 컬럼에 문자열 데이터를 저장한다는 의미입니다.

WHERE KEY='%'s' 는 KEY 값이 일치하는 레코드를 수정한다는 의미입니다.

btn_update_cb() 는 Entry에 입력된 데이터를 현재 선택된 레코드에 저장하는 함수입니다.

예제를 다시 실행시켜 봅시다. List 항목 중에서 하나를 선택하고 Entry에 데이터를 변경한 다음, Update 버튼을 누릅니다. List 위젯에 데이터가 변경됩니다. DB에 저장된 것입니다.



5) 레코드 삭제

3번째 Button을 추가해서 사용자가 Button을 누르면 현재 선택된 레코드를 삭제하는 기능을 구현해 봅시다. create_base_gui() 함수에 Button 생성 코드를 추가합니다.

```

/* Button-Update */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Update");
evas_object_smart_callback_add(btn, "clicked", btn_update_cb, ad);
my_table_pack(table, btn, 1, 1, 1, 1);

/* Button-Del */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Del");
evas_object_smart_callback_add(btn, "clicked", btn_del_cb, ad);
my_table_pack(table, btn, 2, 1, 1, 1);

/* List */
ad->list = elm_list_add(ad->conform);
elm_list_mode_set(ad->list, ELM_LIST_COMPRESS);
elm_list_go(ad->list);
my_table_pack(table, ad->list, 0, 2, 3, 8);

```

그런 다음 create_base_gui() 함수 위에 2개의 새로운 함수를 추가합니다.

```

static int
DelRecord(appdata_s *ad)
{
    char sql[256];
    char *ErrMsg;
    sprintf(sql, 256, "DELETE FROM ReportCard WHERE KEY=%s;", ad->current_key);

    int ret = sqlite3_exec(ad->db, sql, NULL, 0, &ErrMsg);
    return ret;
}

static void

```

```

btn_del_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    DelRecord(ad);

    read_db(ad);
}

```

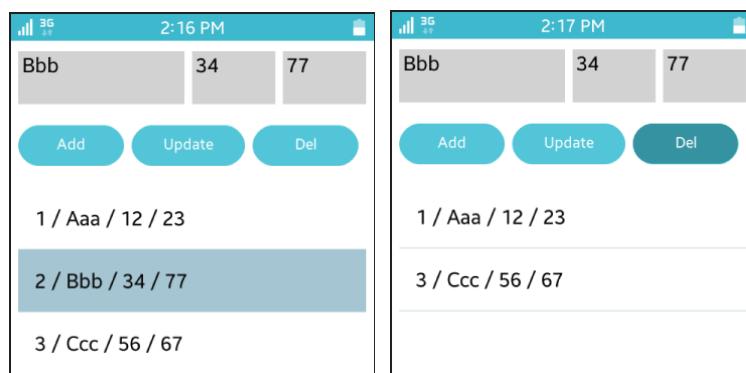
DelRecord() 는 DB에 저장된 레코드를 삭제하는 함수입니다.

쿼리문에서 'DELETE FROM ReportCard' 는 ReportCard 테이블에서 데이터를 삭제한다는 의미입니다.

WHERE KEY=%s 는 Key 값이 일치하는 레코드를 삭제한다는 의미입니다.

btn_del_cb() 는 사용자가 Del 버튼을 눌렀을 때 현재 선택된 List 항목을 삭제하는 함수입니다.

예제를 다시 실행시켜 봅시다. 몇가지 항목을 더 추가한 다음, List 항목 중에서 하나를 선택하고 Del 버튼을 누르면 해당 항목이 사라집니다.



6) 관련 API

int sqlite3_exec(sqlite3*, char *, int (*callback), void *, char **) : SQL 쿼리문을 실행하는 API. 파라미터는 순서대로 SQLite 객체, 쿼리문, 콜백 함수명, 사용자 데이터, 에러메시지 반환 입니다.

int sqlite3_shutdown() : DB를 닫는 API.

int sqlite3_config(int, ...) : DB 속성을 지정하는 API.
SQLITE_CONFIG_URI를 전달하면 DB 파일에 데이터를 저장할 수 있습니다.

int sqlite3_initialize() : DB를 초기화하는 API.

char *app_get_data_path() : /data 폴더의 절대 경로를 반환하는 API. /res 폴더에는 Write가 허용되지 않기 때문에 /data 폴더에 DB 파일을 생성하겠습니다.

char *strncat (char *, char *, size_t) : 최대 길이를 지정해서 문자열 뒤에 새로운 문자열을 추가하는 API. 파라미터는 순서대로 원본 문자열 배열, 추가될 문자열 데이터, 최대 길이입니다.

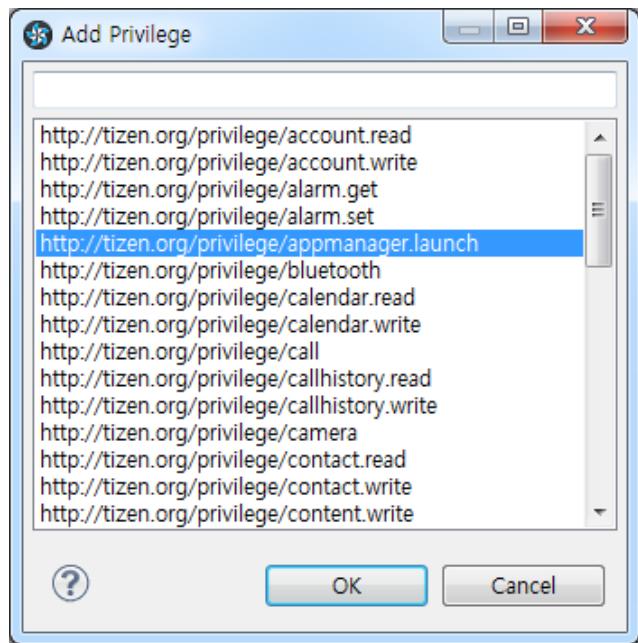
int sqlite3_open(char *, sqlite3 **) : DB 파일을 Open 하는 API. DB 파일이 존재하지 않으면 생성합니다. 1번째 파라미터에 파일 경로를 전달하면, 2번째 파라미터에서 DB 객체를 반환합니다.

62. AppControl로 외부앱 호출하기

개발해야 하는 앱에 이미지 뷰어 혹은 카메라 기능이 필요할 때 전체 기능을 하드코딩으로 구현하기에는 너무 힘든 작업일 것입니다. 타이젠 플랫폼에 기본적으로 설치되어 있는 앱들을 AppControl이라고 하며, 다른 앱에서 AppControl을 Loading 할 수 있습니다. 기본 앱이 아니더라도 패키지명만 알고 있다면 Loading 할 수 있습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 AppControlEx 으로 지정합니다. 외부 앱을 실행하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/appmanager.launch> 을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



동일한 과정을 반복해서 다음 2개의 사용자 권한을 추가합니다.

- http://tizen.org/privilege/internet
- http://tizen.org/privilege/email

저장하고 아래쪽 탭버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.appcontrolex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application      appid="org.example.appcontrolex"      exec="appcontrolex"
multiple="false" nodisplay="false" taskmanage="true" type="capp">
        <label>appcontrolex</label>
        <icon>appcontrolex.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/internet</privilege>
        <privilege>http://tizen.org/privilege/appmanager.launch</privilege>
```

```
<privilege>http://tizen.org/privilege/email</privilege>
</privileges>
</manifest>
```

2) 예제 Loading

src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수, 그리고 구조체를 추가합니다.

```
#include "appcontrolex.h"
#include <app.h>
#include <app_control.h>

#define FM_PHONE_FOLDER "/opt/usr/media"

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
} appdata_s;
```

app.h 와 app_control.h 은 AppControl을 사용하기 위한 라이브러리 헤더파일입니다.

"/opt/usr/media" 는 공유폴더 루트 경로입니다.

우리가 이제까지 만들어본 예제 중에서 HelloWorld 예제를 Loading 시켜 보겠습니다. create_base_gui() 함수에 Box, Button 생성 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{ /* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{ /* child object - indent to how relationship */
/* Label*/
ad->label = elm_label_add(ad->win);
elm_object_text_set(ad->label, "<align=center>Hello Tizen</>");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

/* Button-1 */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Sample App");
evas_object_smart_callback_add(btn, "clicked", btn_sample_app_cb, ad);
/* expand both horiz and vert, fill horiz and vert */
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
}

}

```

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

}
```

create_base_gui() 함수 위에 2개의 새로운 함수를 생성합니다.

```
static void
btn_sample_app_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    app_control_h app_control;
    app_control_create(&app_control);
    app_control_set_operation(app_control, APP_CONTROL_OPERATION_DEFAULT);
    app_control_set_app_id (app_control, "org.example.helloworld");

    if      (app_control_send_launch_request(app_control,      NULL,      NULL) == APP_CONTROL_ERROR_NONE)
        dlog_print(DLOG_INFO, "tag", "Succeeded to launch a Helloworld app.");
    else
        dlog_print(DLOG_INFO, "tag", "Failed to launch a calculator app.");

    app_control_destroy(app_control);
}

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
```

```

        * "default" frame style */
        elm_object_style_set(frame, "pad_medium");
        /* set the input weight/aling on the frame insted of the child */
        evas_object_size_hint_weight_set(frame, h_weight, v_weight);
        evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}

```

Btn_sample_app_cb() 는 Button 콜백 함수 입니다.

app_control_h 는 AppControl 구조체입니다.

app_control_create(app_control_h *) 는 AppControl 객체를 생성하는 API입니다.

app_control_set_operation(app_control_h, char *) 는 AppControl의 역할을 지정하는 API입니다. 외부 앱을 Loading 할 때는 APP_CONTROL_OPERATION_DEFAULT를 전달하면 됩니다.

app_control_set_app_id(app_control_h, char *) 는 Loading 할 예제의

패키지명을 AppControl에 지정하는 API입니다. HelloWorld 예제의 패키지명은 "org.example.helloworld"입니다.

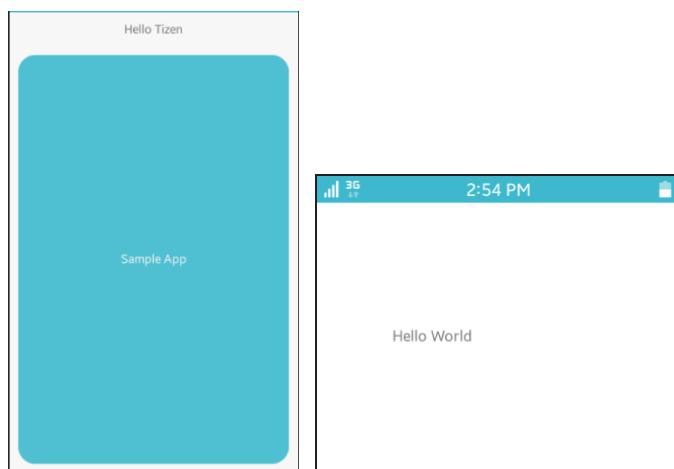
app_control_send_launch_request(app_control_h, app_control_reply_cb, void *) 는 AppControl을 실행하는 API입니다.

app_control_destroy(app_control_h) 는 AppControl 객체를 삭제하는 API입니다.

my_box_pack() 은 box에 위젯을 추가하는 함수입니다.

예제를 실행시키기 전에 에뮬레이터에 HelloWorld 예제가 설치되어 있는지 확인해 봅시다. 만약 설치되어 있지 않다면 HelloWorld 라는 이름의 소스프로젝트를 생성해서 에뮬레이터에 설치합니다.

HelloWorld 예제를 에뮬레이터에 설치했으면 AppControlEx 예제를 빌드하고 실행시켜 봅시다. Button을 누르면 HelloWorld 예제가 실행됩니다.



3) 다른 앱에 데이터 전달

HelloWorld 예제를 Loading 할 때 데이터를 전달해 보겠습니다.
appcontrollex.c 파일의 btn_sample_app_cb() 함수에 새로운 코드를 추가합니다.

```
[appcontrollex.c]-----  
static void  
btn_sample_app_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
  
    app_control_h app_control;  
    app_control_create(&app_control);  
    app_control_set_operation(app_control, APP_CONTROL_OPERATION_DEFAULT);  
    app_control_add_extra_data(app_control, "pet", "dog");  
    app_control_add_extra_data(app_control, "dessert", "juice");  
    app_control_set_app_id (app_control, "org.example.helloworld");  
  
    if      (app_control_send_launch_request(app_control,      NULL,      NULL) ==  
APP_CONTROL_ERROR_NONE)  
        dlog_print(DLOG_INFO, "tag", "Succeeded to launch a Helloworld app.");  
    else  
        dlog_print(DLOG_INFO, "tag", "Failed to launch a calculator app.");  
  
    app_control_destroy(app_control);  
}
```

app_control_add_extra_data(app_control_h, char *, char *) 는
AppControl에 데이터를 추가하는 API입니다. 파라미터는 순서대로
AppControl 객체, Key 값, 텍스트 데이터 입니다.

보내는 쪽은 이것으로 구현 되었고, 받는 쪽을 구현할 차례입니다.

HelloWorld 예제의 소스 파일(/src/helloworld.c)을 열고 위쪽에 전역변수를 추가합니다.

```
[helloworld.c]-----  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
} appdata_s;  
  
char recv_data[100];  
-----
```

AppControl의 추가 데이터 개수 만큼 이벤트 함수가 호출됩니다. 그래서 전체 데이터를 저장하기 위해서 전역변수를 선언하였습니다.

소스파일 아래쪽으로 이동하면 app_control() 이라는 함수가 있습니다. 이것이 AppControl 이벤트 함수입니다. 새로운 코드와, 새로운 함수를 추가합니다.

```
[helloworld.c]-----  
bool _app_control_extra_data_cb(app_control_h app_control, const char *key, void  
*data)  
{  
    int ret;  
    char *value;  
  
    ret = app_control_get_extra_data(app_control, key, &value);  
    strcat(recv_data, key);  
    strcat(recv_data, ":" );  
    strcat(recv_data, value);  
    strcat(recv_data, " / ");  
  
    appdata_s *ad = data;
```

```
elm_object_text_set(ad->label, recv_data);
return true;
}

static void
app_control(app_control_h app_control, void *data)
{
    app_control_FOREACH_EXTRA_DATA(app_control, _app_control_extra_data_cb, data);
}
```

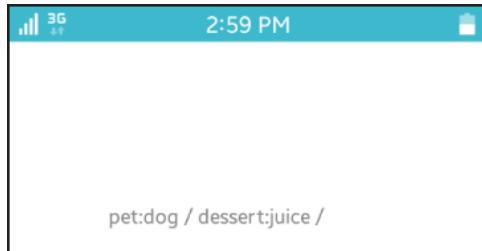
_app_control_extra_data_cb() 는 개별 데이터를 수신하는 콜백 함수입니다. AppControlEx 예제에서 2개의 데이터를 전달했기 때문에 이 함수도 2번 호출됩니다.

app_control_get_extra_data(app_control_h, char *, char **) 는 AppControl에서 데이터를 추출하는 API입니다. 2번째 파라미터에 Key를 전달하면 3번째 파라미터에서 데이터를 반환합니다.

app_control() 는 AppControl 객체가 수신되었을 때 실행되는 콜백 함수입니다.

app_control_FOREACH_EXTRA_DATA(app_control_h, app_control_extra_data_cb, void *) 는 AppControl 객체에 저장된 개별 데이터 처리 함수를 지정하는 API입니다.

HelloWorld 예제를 먼저 에뮬레이터에 설치한 다음, AppControlEx 예제를 실행시켜 보겠습니다. Button을 누르면 HelloWorld 예제가 실행되고 AppControlEx 예제에서 전달한 데이터가 Label 위젯에 표시됩니다.



4) 카메라 AppControl

기본 앱 중에서 카메라 앱을 Loading 하는 방법을 알아보겠습니다.
AppControlEx 예제의 소스파일로 이동해서 create_base_gui() 함수
끝부분에서 새로운 Button을 생성합니다.

```
[appcontrollex.c]-----  
/* Button-1 */  
Evas_Object *btn = elm_button_add(ad->conform);  
elm_object_text_set(btn, "Sample App");  
evas_object_smart_callback_add(btn, "clicked", btn_sample_app_cb, ad);  
/* expand both horiz and vert, fill horiz and vert */  
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);  
  
/* Button-2 */  
btn = elm_button_add(ad->conform);  
elm_object_text_set(btn, "Camera");  
evas_object_smart_callback_add(btn, "clicked", btn_camera_cb, ad);  
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);  
}  
}  
-----
```

2번째 Button의 콜백 함수를 만들어 보겠습니다. create_base_gui() 함수
위에 새로운 함수를 추가합니다.

```

static void
btn_camera_cb(void *data, Evas_Object *obj, void *event_info)
{
    app_control_h app_control;

    app_control_create(&app_control);
    app_control_set_operation(app_control,
APP_CONTROL_OPERATION_CREATE_CONTENT);
    app_control_set_mime(app_control, "image/jpg");
    if      (app_control_send_launch_request(app_control,      NULL,      NULL) ==
APP_CONTROL_ERROR_NONE)
        dlog_print(DLOG_INFO, "tag", "Succeeded to launch camera app.");
    else
        dlog_print(DLOG_INFO, "tag", "Failed to launch camera app.");

    app_control_destroy(app_control);
}

```

`app_control_set_operation(app_control_h, char *)` 는 AppControl의 역할을 지정하는 API입니다.

사진을 촬영하면 이미지 파일이 생성되기 때문에 APP_CONTROL_OPERATION_CREATE_CONTENT를 전달하면 됩니다.

`app_control_set_mime(app_control_h, char *)` 는 MIME 형식을 지정하는 API입니다. 카메라 앱도 여러 종류가 설치되어 있을수 있기 때문에 특정 앱을 지정하는 것이 아니라 MIME 형식으로 지정해야 합니다.

예제를 다시 실행하고 2번째 Button을 눌러봅시다. 카메라 앱이 실행됩니다. 에뮬레이터에서 실행되지 않을 때는 폰에서 테스트 하면 됩니다.



5) E-Mail AppControl

이번에는 기본 앱 중에서 E-Mail 앱을 Loading 하는 방법을 알아보겠습니다. `create_base_gui()` 함수 끝부분에서 새로운 Button을 생성합니다.

```

/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Camera");
evas_object_smart_callback_add(btn, "clicked", btn_camera_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "E-mail");
evas_object_smart_callback_add(btn, "clicked", btn_email_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
}

```

3번째 Button의 콜백 함수를 만들어 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void
btn_email_cb(void *data, Evas_Object *obj, void *event_info)
{
    app_control_h app_control;
    char *mail_address = "topofsan@naver.com";
    char *subject = "Tutorial message title";
    char *message = "Tutorial message content.";

    app_control_create(&app_control);
    app_control_set_operation(app_control, APP_CONTROL_OPERATION_COMPOSE);
    app_control_set_app_id(app_control, "email-composer-efl");
    app_control_add_extra_data(app_control, APP_CONTROL_DATA_TEXT, message);
    app_control_add_extra_data(app_control, APP_CONTROL_DATA_TO, mail_address);
    app_control_add_extra_data(app_control, APP_CONTROL_DATA SUBJECT, subject);
    if      (app_control_send_launch_request(app_control,      NULL,      NULL) == APP_CONTROL_ERROR_NONE)
        dlog_print(DLOG_INFO, "tag", "Succeeded to launch e-mail app.");
    else
        dlog_print(DLOG_INFO, "tag", "Failed to launch e-mail app.");

    app_control_destroy(app_control);
}
```

`app_control_set_operation(app_control_h, char *)` 는 AppControl의 역할을 지정하는 API입니다. E-Mail 발송에는 `APP_CONTROL_OPERATION_COMPOSE`를 전달하면 됩니다.

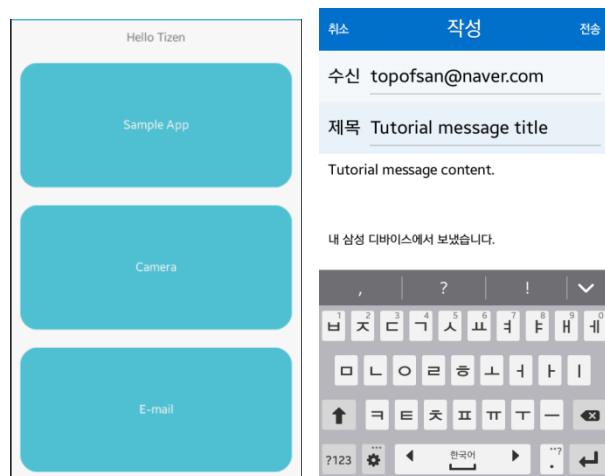
`app_control_set_app_id(app_control_h, char *)` 는 Loading 할 예제의 패키지명을 AppControl에 지정하는 API 입니다. E-Mail 발송은 "email-

"composer-efl" 입니다.

app_control_add_extra_data(app_control_h, char *, char *) 는 AppControl에 데이터를 추가하는 API입니다. 파라미터는 순서대로 AppControl 객체, Key 값, 텍스트 데이터입니다. E-Mail의 경우에 사용되는 Key는 다음과 같습니다.

- APP_CONTROL_DATA_TEXT : 메일 본문 내용
- APP_CONTROL_DATA_TO : 받는 사람 주소
- APP_CONTROL_DATA_SUBJECT : 메일 제목

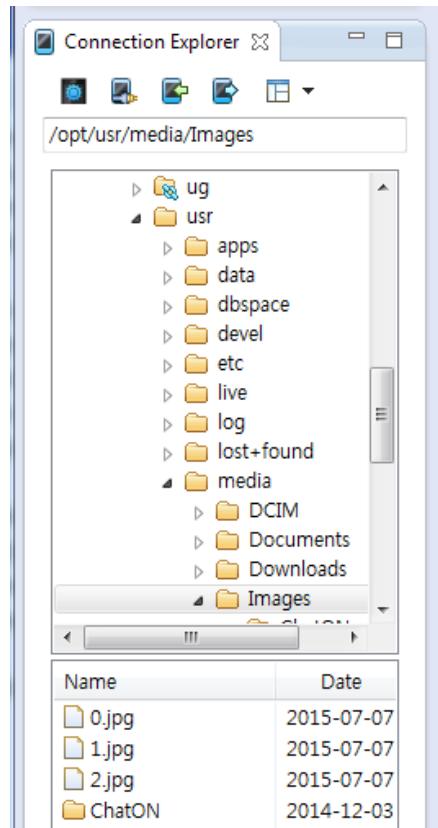
예제를 다시 실행하고 3번째 Button을 눌러 봅시다. E-Mail 앱이 실행됩니다. 에뮬레이터에서 실행되지 않을 때는 폰에서 테스트 하면 됩니다.



6) Image Viewer AppControl

이번에는 기본 앱 중에서 Image Viewer 앱을 Loading 하는 방법을 알아보겠습니다. Image Viewer 앱에 이미지를 표시하기 위해서는 이미지 파일이 필요합니다. 에뮬레이터 /opt/usr/media/Images 폴더에 0.jpg

파일이 존재하는지 확인해 봅시다. 만약 파일이 없다면 부록 /Image 폴더에서 0.jpg 파일을 단말로 복사하겠습니다. 이클립스 Connection Explorer에서 /opt/usr/media/Images 폴더에 드래그 하면 됩니다.



create_base_gui() 함수 끝부분에서 새로운 Button을 생성합니다.

```
/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "E-mail");
evas_object_smart_callback_add(btn, "clicked", btn_email_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Button-4 */
btn = elm_button_add(ad->conform);
```

```

        elm_object_text_set(btn, "Image Viewer");
        evas_object_smart_callback_add(btn, "clicked", btn_image_viewer_cb,
ad);
        my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
    }
}

```

4번째 Button의 콜백 함수를 만들어 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```

static void
btn_image_viewer_cb(void *data, Evas_Object *obj, void *event_info)
{
    app_control_h app_control;
    char buf[PATH_MAX];
    strcat(buf, FM_PHONE_FOLDER);
    strcat(buf, "/Images/0.jpg");

    app_control_create(&app_control);
    app_control_set_operation(app_control, APP_CONTROL_OPERATION_VIEW);
    app_control_set_uri(app_control, buf);
    app_control_set_mime(app_control, "image/*");

    if (app_control_send_launch_request(app_control, NULL, NULL) ==
APP_CONTROL_ERROR_NONE)
        dlog_print(DLOG_INFO, "tag", "Succeeded to launch Image Viewer app.");
    else
        dlog_print(DLOG_INFO, "tag", "Failed to launch Image Viewer app.");

    app_control_destroy(app_control);
}

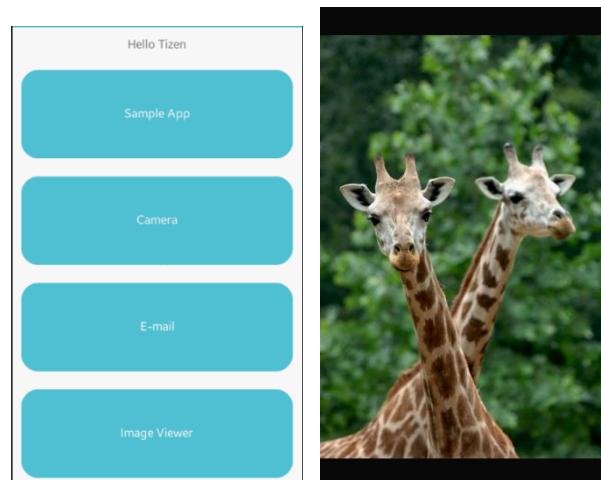
```

`app_control_set_operation(app_control_h, char *)` 는 AppControl의 역할을 지정하는 API입니다. Image Viewer 앱을 Loading 할 때는 APP_CONTROL_OPERATION_VIEW를 전달하면 됩니다.

`app_control_set_uri(app_control_h, char *)` 는 컨텐츠의 경로를 지정하는 API입니다. 이미지 파일의 경로를 지정하였습니다.

`app_control_set_mime(app_control_h, char *)` 는 MIME 형식을 지정하는 API입니다. Image Viewer 앱도 여러 종류가 설치되어 있을수 있기 때문에 특정 앱을 지정하는 것이 아니라 MIME 형식으로 지정해야 합니다.

예제를 다시 실행하고 4번째 Button을 눌러봅시다. Image View가 실행됩니다.



7) 웹 브라우저 AppControl

이번에는 기본 앱 중에서 웹 브라우저 앱을 Loading 하는 방법을 알아보겠습니다. `create_base_gui()` 함수 끝부분에서 새로운 Button을 생성합니다.

```

/* Button-4 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Image Viewer");
evas_object_smart_callback_add(btn, "clicked", btn_image_viewer_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Button-5 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Web browser");
evas_object_smart_callback_add(btn, "clicked", btn_web_browser_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
}

```

5번째 Button의 콜백 함수를 만들어 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```

static void
btn_web_browser_cb(void *data, Evas_Object *obj, void *event_info)
{
    app_control_h app_control;

    app_control_create(&app_control);
    app_control_set_operation(app_control, APP_CONTROL_OPERATION_DEFAULT);
    app_control_set_app_id(app_control, "com.samsung.browser");
    app_control_set_uri(app_control, "www.tizen.org");
    if      (app_control_send_launch_request(app_control,      NULL,      NULL) == APP_CONTROL_ERROR_NONE)
        dlog_print(DLOG_INFO, "tag", "Succeeded to launch browser app.");
    else
        dlog_print(DLOG_INFO, "tag", "Failed to launch browser app.");
    app_control_destroy(app_control);
}

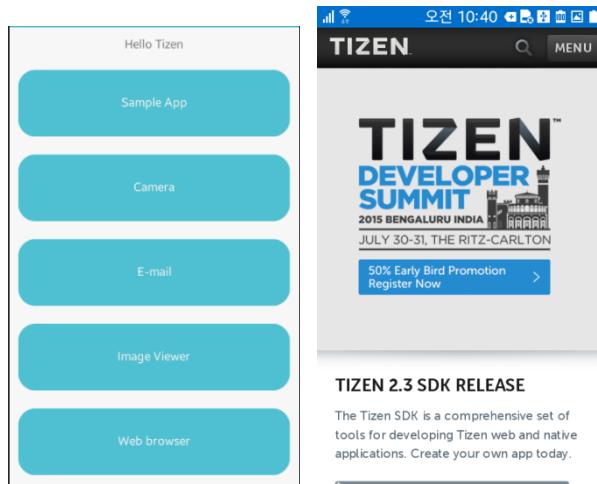
```

```
}
```

app_control_set_app_id(app_control_h, char *) 는 Loading 할 예제의 패키지명을 AppControl에 지정하는 API입니다. 기본 웹 브라우저 앱의 패키지명은 "com.samsung.browser"입니다.

app_control_set_uri(app_control_h, char *) 는 컨텐츠의 경로를 지정하는 API입니다. Tizen 웹 사이트 주소를 지정하였습니다.

예제를 다시 실행하고 5번째 Button을 눌러봅시다. 웹 브라우저가 실행됩니다.



8) 관련 API

int app_control_create(app_control_h *) : AppControl 객체 생성.

int app_control_set_operation(app_control_h, char *) : AppControl의 역할을 지정하는 API. 외부 앱을 Loading 할 때는 APP_CONTROL_OPERATION_DEFAULT를 전달하면 됩니다.

int app_control_set_app_id(app_control_h, char *) : Loading 할 예제의

패키지명을 AppControl에 지정하는 API. HelloWorld 예제의 패키지명은 "org.example.helloworld"입니다.

`int app_control_send_launch_request(app_control_h, app_control_reply_cb, void *)` : AppControl을 실행하는 API.

`int app_control_destroy(app_control_h)` : AppControl 객체 삭제.

`int app_control_add_extra_data(app_control_h, char *, char *)` : AppControl에 데이터를 추가하는 API. 파라미터는 순서대로 AppControl 객체, Key 값, 텍스트 데이터입니다.

`int app_control_get_extra_data(app_control_h, char *, char **)` : AppControl에서 데이터를 추출하는 API. 2번째 파라미터에 Key를 전달하면 3번째 파라미터에서 데이터를 반환합니다.

`int app_control_foreach_extra_data(app_control_h, app_control_extra_data_cb, void *)` : AppControl 객체에 저장된 개별 데이터 처리 함수를 지정하는 API.

`int app_control_set_operation(app_control_h, char *)` : AppControl의 역할을 지정하는 API. 사진을 촬영하면 이미지 파일이 생성되기 때문에 APP_CONTROL_OPERATION_CREATE_CONTENT를 전달하면 됩니다.

`int app_control_set_mime(app_control_h, char *)` : MIME 형식을 지정하는 API.

`int app_control_set_operation(app_control_h, char *)` : AppControl의 역할을 지정하는 API. E-Mail 발송에는 APP_CONTROL_OPERATION_COMPOSE를 전달하면 됩니다. Image Viewer 앱을 Loading 할 때는 APP_CONTROL_OPERATION_VIEW를 전달하면

됩니다.

int app_control_set_app_id(app_control_h, char *) : Loading 할 예제의 패키지명을 AppControl에 지정하는 API. E-Mail 발송은 "email-composer-efl"입니다. 기본 웹 브라우저 앱의 패키지명은 "com.samsung.browser"입니다.

int app_control_add_extra_data(app_control_h, char *, char *) : AppControl에 데이터를 추가하는 API. 파라미터는 순서대로 AppControl 객체, Key 값, 텍스트 데이터입니다. E-Mail의 경우에 사용되는 Key 는 다음과 같습니다.

- APP_CONTROL_DATA_TEXT : 메일 본문 내용
- APP_CONTROL_DATA_TO : 받는 사람 주소
- APP_CONTROL_DATA SUBJECT : 메일 제목

int app_control_set_uri(app_control_h, char *) : 컨텐츠의 경로를 지정하는 API.

63. 서비스 앱 제작하기

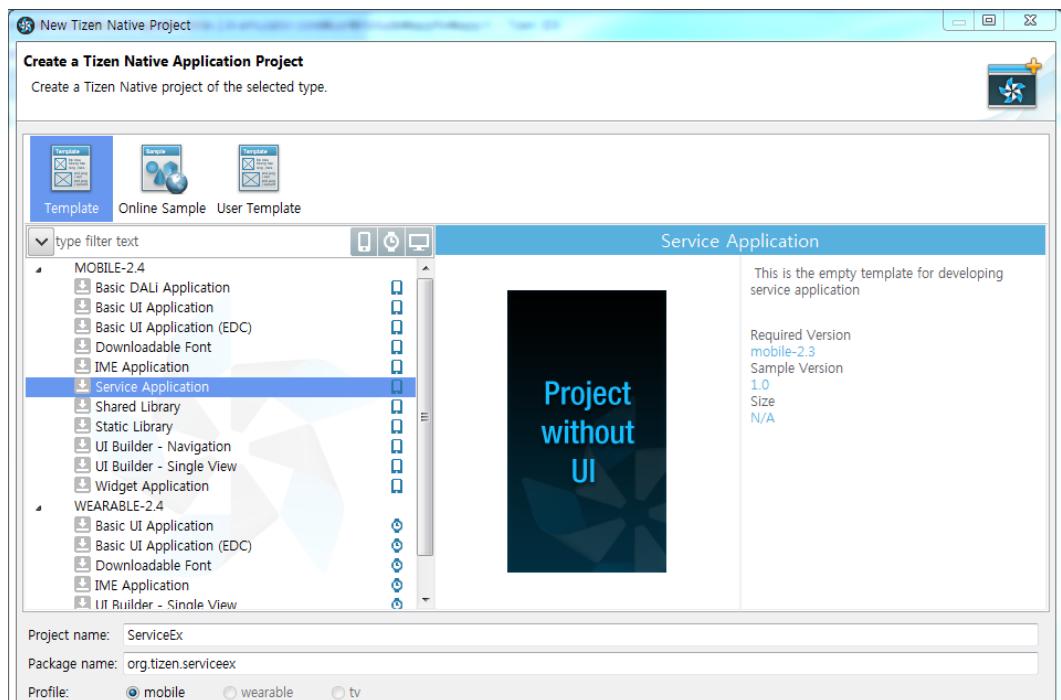
서비스는 사용자에게 보여지는 UI 없이 백그라운드에서 동작하는 앱입니다. 바이러스 백신, 도난방지 기능, 통신 기능 등은 서비스 앱으로 구현하는 경우가 많습니다.

1) 서비스 소스 프로젝트 생성

새로운 소스 프로젝트를 생성하겠습니다. 이클립스 메인 메뉴 [File > New Tizen Native Project]를 선택합니다.

팝업창이 나타나면 [Template > MOBILE-2.x > Service Application]을 선택합니다.

Project name 항목에 ServiceEx 라고 지정하고 Finish 버튼을 누릅니다.



서비스 앱이 실행되면 5초 후에 자동으로 종료되는 기능을 구현해 보겠습니다. src 폴더에 소스파일(~.c)을 열고 define 상수와 변수를 추가합니다.

```
-----  
#include <service_app.h>  
#include "serviceex.h"  
#include <ecore.h>
```

```
Ecore_Timer *timer1;  
int timer_count = 0;
```

Ecore_Timer는 타이머 구조체입니다.

timer_count에는 타이머 이벤트 발생 회수를 저장하겠습니다.

서비스 앱이 실행되면 자동으로 타이머를 구동 시키겠습니다. service_app_create() 는 서비스 앱이 구동되었을 때의 이벤트 함수입니다. service_app_terminate() 는 서비스 앱이 종료될 때의 이벤트 함수입니다. 새로운 코드를 추가합니다.

```
-----  
bool service_app_create(void *data)  
{  
    dlog_print(DLOG_DEBUG, "tag", "%s", __func__);  
    timer_count = 0;  
    timer1 = ecore_timer_add(1.0, timer1_cb, NULL);  
    return true;  
}
```

```
void service_app_terminate(void *data)
```

```
{
```

```
dlog_print(DLOG_DEBUG, "tag", "%s", __func__);
return;
}
```

앱이 실행될 때와 종료될 때에 Log 메시지를 출력합니다. 그리고 앱이 실행될 때는 타이머를 실행합니다.

타이머 이벤트 콜백 함수를 생성하겠습니다. service_app_create() 함수 위에 새로운 함수를 추가합니다.

```
static Eina_Bool
timer1_cb(void *data EINA_UNUSED)
{
    timer_count++;
    char buf[100];
    sprintf(buf, "Count - %d", timer_count);
    dlog_print(DLOG_DEBUG, "tag", "%s - %s", __func__, buf);

    if( timer_count > 5 )
        service_app_exit();
    return ECORE_CALLBACK_RENEW;
}
```

timer1_cb() 는 타이머 이벤트 함수입니다. 1초에 한번씩 실행되어서 Log 메시지를 출력하고 5초가 지나면 자동으로 앱을 종료합니다.

service_app_exit() 는 서비스 앱을 종료하는 API 입니다.

예제를 빌드하고 실행시켜 봅시다. 에뮬레이터에는 아무런 변화가 없습니다. 서비스 앱은 UI 화면이 없기 때문입니다. Log 메시지를 확인해

보겠습니다. Log 패널 아래쪽 콤보박스에 Tag를 선택하고 오른쪽 에디트박스에 tag 라고 입력합니다. 앱 시작 메시지와 6번의 카운트, 그리고 앱 종료 메시지가 출력됩니다.

The screenshot shows the Eclipse IDE interface with the 'Log' tab selected. A search filter 'tag' is applied, resulting in the following log entries:

Time	Level	Pid	Tid	Tag	Message
07-07 11:51:36.583	Debug	2537	2537	tag	service_app_create
07-07 11:51:37.593	Debug	2537	2537	tag	timer1_cb - Count - 1
07-07 11:51:38.583	Debug	2537	2537	tag	timer1_cb - Count - 2
07-07 11:51:39.593	Debug	2537	2537	tag	timer1_cb - Count - 3
07-07 11:51:40.593	Debug	2537	2537	tag	timer1_cb - Count - 4
07-07 11:51:41.583	Debug	2537	2537	tag	timer1_cb - Count - 5
07-07 11:51:42.593	Debug	2537	2537	tag	timer1_cb - Count - 6
07-07 11:51:42.593	Debug	2537	2537	tag	service_app_terminate

2) 외부 앱에서 서비스 앱으로 이벤트 전달하기

외부의 앱에서 이벤트를 전달해서 서비스 앱을 종료하는 기능을 구현해 보겠습니다. 방법은 AppControlEx 예제에서 HelloWorld 예제에 데이터를 전달할 때와 동일합니다.

소스 파일에 service_app_control()라는 함수가 있는데 AppControl 이벤트 함수입니다. 이 함수에 새로운 코드를 추가하고, 그 위에 새로운 함수를 생성합니다.

```
bool _app_control_extra_data_cb(app_control_h app_control, const char *key, void *data)
{
    int ret;
    char *value;
```

```

ret = app_control_get_extra_data(app_control, key, &value);
dlog_print(DLOG_DEBUG, "tag", "%s - %s : %s", __func__, key, value);
if( strcmp(key, "dessert") == 0 && strcmp(value, "juice") == 0 )
{
    dlog_print(DLOG_DEBUG, "tag", "Close message received");
    service_app_exit();
}
return true;
}

void service_app_control(app_control_h app_control, void *data)
{
    dlog_print(DLOG_DEBUG, "tag", "%s", __func__);
    app_control_FOREACH_EXTRA_DATA(app_control, _app_control_extra_data_cb, NULL);
    return;
}

```

`_app_control_extra_data_cb()` 는 개별 데이터를 수신하는 콜백 함수입니다. AppControl에 2개의 데이터가 저장되어 있다면 이 함수도 2번 호출됩니다.

`app_control_get_extra_data(app_control_h, char *, char **)` 는 AppControl에서 데이터를 추출하는 API입니다. 2번째 파라미터에 Key를 전달하면 3번째 파라미터에서 데이터를 반환합니다.

`service_app_control()` 는 AppControl 객체가 수신되었을 때 실행되는 콜백 함수입니다.

`app_control_FOREACH_EXTRA_DATA(app_control_h, app_control_extra_data_cb, void *)` 는 AppControl 객체에 저장된 개별 데이터 처리 함수를 지정하는 API입니다.

timer1_cb() 함수의 코드를 다음과 같이 수정합니다. 5초만 지나면 자동으로 서비스 앱이 종료되기 때문에 시간을 더 길게 지정합니다.

```
static Eina_Bool  
timer1_cb(void *data EINA_UNUSED)  
{  
    timer_count++;  
    char buf[100];  
    sprintf(buf, "Count - %d", timer_count);  
    dlog_print(DLOG_DEBUG, "tag", "%s - %s", __func__, buf);  
  
    //if( timer_count > 5 )  
    if( timer_count > 50 )  
        service_app_exit();  
    return ECORE_CALLBACK_RENEW;  
}
```

외부 앱에서 서비스 앱을 호출하는 기능을 구현해 보겠습니다. 이전 시간에 만들었던 AppControlEx 예제의 소스파일을 엽니다. 그리고 create_base_gui() 함수 끝부분에 Button 생성 코드를 추가합니다.

```
[appcontrollex.c]—————  
/* Button-5 */  
btn = elm_button_add(ad->conform);  
elm_object_text_set(btn, "Web browser");  
evas_object_smart_callback_add(btn, "clicked", btn_web_browser_cb, ad);  
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);  
  
/* Button-6 */  
btn = elm_button_add(ad->conform);
```

```

        elm_object_text_set(btn, "Service close");
        evas_object_smart_callback_add(btn, "clicked", btn_service_close_cb, ad);
        my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
    }
}

```

create_base_gui() 함수 위에 Button 콜백 함수를 생성합니다.

```

static void
btn_service_close_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;

    app_control_h app_control;
    app_control_create(&app_control);
    app_control_set_operation(app_control, APP_CONTROL_OPERATION_DEFAULT);
    app_control_add_extra_data(app_control, "pet", "dog");
    app_control_add_extra_data(app_control, "dessert", "juice");
    app_control_set_app_id (app_control, "org.example.serviceex");

    if      (app_control_send_launch_request(app_control,      NULL,      NULL) == APP_CONTROL_ERROR_NONE)
        dlog_print(DLOG_INFO, "tag", "Succeeded to launch a Service app.");
    else
        dlog_print(DLOG_INFO, "tag", "Failed to launch a Service app.");

    app_control_destroy(app_control);
}

```

app_control_h는 AppControl 구조체입니다.

app_control_create(app_control_h *) 는 AppControl 객체를 생성하는

API입니다.

`app_control_set_operation(app_control_h, char *)` 는 AppControl의 역할을 지정하는 API입니다. 외부 앱을 Loading 할 때는 `APP_CONTROL_OPERATION_DEFAULT`를 전달하면 됩니다.

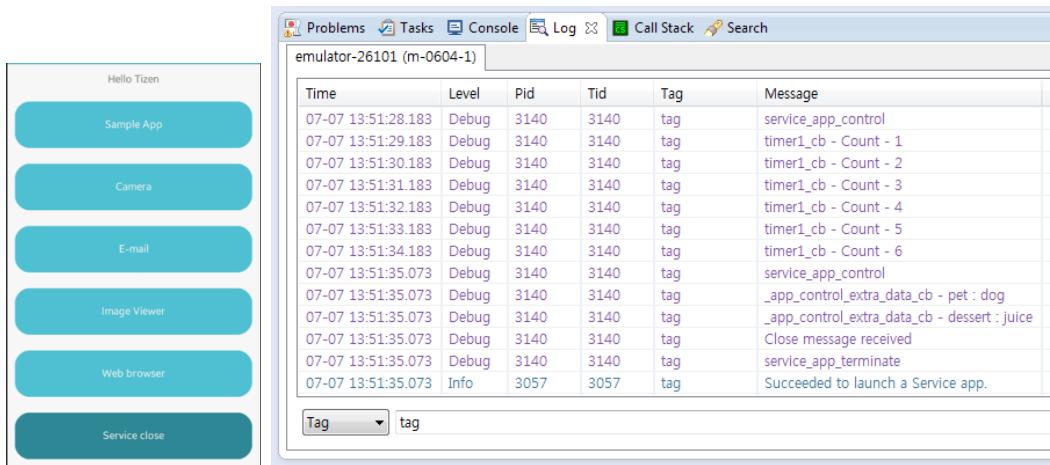
`app_control_add_extra_data(app_control_h, char *, char *)` 는 AppControl에 데이터를 추가하는 API입니다. 파라미터는 순서대로 AppControl 객체, Key 값, 텍스트 데이터 입니다.

`app_control_set_app_id(app_control_h, char *)` 는 Loading 할 예제의 패키지명을 AppControl에 지정하는 API입니다.

`app_control_send_launch_request(app_control_h, app_control_reply_cb, void *)` 는 AppControl을 실행하는 API입니다.

`app_control_destroy(app_control_h)` 는 AppControl 객체를 삭제하는 API입니다.

서비스 앱을 먼저 실행한 후에 AppControlEx 예제를 실행 실행합니다. 6번째 Button을 누르면 서비스 앱에서 Log 메시지가 중단됩니다. 서비스 앱이 종료된 것입니다.



3) 관련 API

app_control_h : AppControl 구조체.

int app_control_create(app_control_h *) : AppControl 객체를 생성하는 API.

int app_control_set_operation(app_control_h, char *) : AppControl의 역할을 지정하는 API. 외부 앱을 Loading 할 때는 APP_CONTROL_OPERATION_DEFAULT를 전달하면 됩니다.

int app_control_add_extra_data(app_control_h, char *, char *) : AppControl에 데이터를 추가하는 API. 파라미터는 순서대로 AppControl 객체, Key 값, 텍스트 데이터입니다.

int app_control_set_app_id(app_control_h, char *) : Loading 할 예제의 패키지명을 AppControl에 지정하는 API.

int app_control_send_launch_request(app_control_h, app_control_reply_cb,

`void *)` : AppControl을 실행하는 API.

`int app_control_destroy(app_control_h)` : AppControl 객체를 삭제하는 API.

`int app_control_get_extra_data(app_control_h, char *, char **)` : AppControl에서 데이터를 추출하는 API. 2번째 파라미터에 Key를 전달하면 3번째 파라미터에서 데이터를 반환합니다.

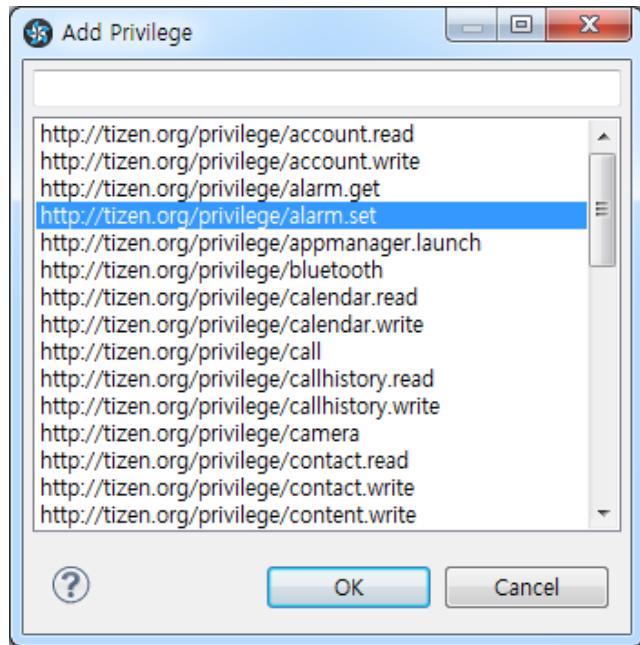
`int app_control_foreach_extra_data(app_control_h, app_control_extra_data_cb, void *)` : AppControl 객체에 저장된 개별 데이터 처리 함수를 지정하는 API.

64. Alarm – 지정된 시간에 앱 실행하기

모닝콜 같은 알람 앱을 개발할 때는 앱이 종료 상태일 때도 지정된 시간이 되면 자동으로 앱이 실행되어야 합니다. Alarm을 사용하면 정해진 시간에 앱을 실행되도록 할 수 있습니다. 타이머 기능도 있어서 일정 시간이 경과한 후에 앱을 실행시킬 수도 있습니다. 사용방법은 AppControl과 유사합니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 Alarm으로 지정합니다. Alarm을 사용하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 텁버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/alarm.set>을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



저장하고 아래쪽 탭버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.alarm" version="1.0.0">
    <profile name="mobile"/>
    <ui-application      appid="org.example.alarm"      exec="alarm"      multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>alarm</label>
        <icon>alarm.png</icon>
    </ui-application>
    <privileges>
        <b><privilege>http://tizen.org/privilege/alarm.set</privilege></b>
    </privileges>
</manifest>
```

2) Timer Alarm 시작

Button을 누르면 3초 후에 HelloWorld 예제가 실행되는 기능을 구현해 보겠습니다. 에뮬레이터에 HelloWorld 예제가 설치되어 있지 않다면 HelloWorld 예제를 먼저 설치합니다. src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
-----  
#include "alarm.h"  
#include <app_alarm.h>  
#include <time.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label;  
    int timer_id;  
    int date_id;  
} appdata_s;  
-----
```

app_alarm.h는 Alarm을 사용하기 위한 라이브러리 헤더파일입니다.

time.h는 시간 관련 API를 사용하기 위한 라이브러리 헤더파일입니다.

timer_id에는 Timer Alarm의 ID를 저장하겠습니다. 이것을 이용해서 Alarm을 종료할 수 있습니다.

date_id에는 Date Alarm의 ID를 저장하겠습니다. 이것을 이용해서 Alarm을 종료할 수 있습니다.

create_base_gui() 함수 위에 새로운 함수를 추가합니다. Box에 위젯을 추가하는 함수입니다.

```

static void
my_box_pack(Evas_Object *box, Evas_Object *child,
            double h_weight, double v_weight, double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL, EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}

```

create_base_gui() 함수 끝부분에 Button 생성 코드를 추가합니다.

```

/* Conformant */

```

```

ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{

/* child object - indent to how relationship */
/* A box to put things in vertically - default mode for box */
Evas_Object *box = elm_box_add(ad->win);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* child object - indent to how relationship */
/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello Tizen</>");
/* expand horizontally but not vertically, and fill horiz,
* align center vertically */
my_box_pack(box, ad->label, 1.0, 0.0, -1.0, 0.5);

/* Button-1 */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Start Timer Alarm");
evas_object_smart_callback_add(btn, "clicked", btn_start_timer_cb, ad);
/* epand both horiz and vert, fill horiz and vert */
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);
}

}

```

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

}
```

create_base_gui() 함수 위에 Button 콜백 함수를 생성하겠습니다.

```
static void
btn_start_timer_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    int ret;
    int DELAY = 3;
    int REMIND = 0;

    app_control_h app_control = NULL;
    ret = app_control_create(&app_control);
    ret = app_control_set_operation(app_control, APP_CONTROL_OPERATION_DEFAULT);
    ret = app_control_set_app_id (app_control, "org.example.helloworld");
    ret = alarm_schedule_after_delay(app_control, DELAY, REMIND, &ad->timer_id);
    dlog_print(DLOG_DEBUG, "tag", "result = %d", ret);
    elm_object_text_set(ad->label, "Timer Alarm Started");
}
```

app_control_h 는 AppControl 구조체입니다.

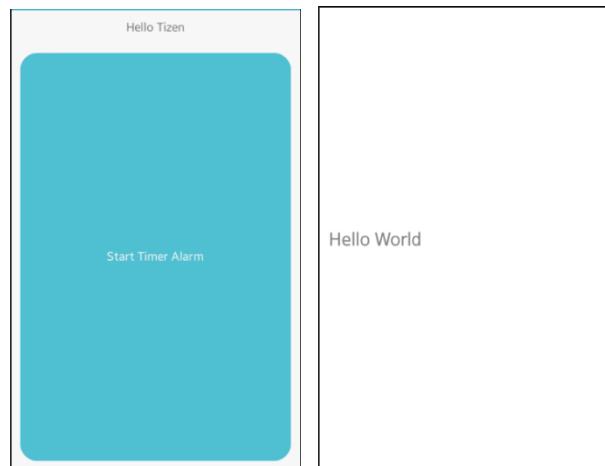
app_control_create(app_control_h *) 는 AppControl 객체를 생성하는 API입니다.

app_control_set_operation(app_control_h, char *) 는 AppControl의 역할을 지정하는 API입니다. 외부 앱을 Loading 할 때는 APP_CONTROL_OPERATION_DEFAULT를 전달하면 됩니다.

`app_control_set_app_id(app_control_h, char *)` 는 Loading 할 예제의 패키지명을 AppControl에 지정하는 API입니다. HelloWorld 예제의 패키지명은 "org.example.helloworld"입니다.

`alarm_schedule_after_delay(app_control_h, int, int, int *)` 는 일정 시간이 경과한 후에 AppControl 이벤트를 전달하는 API입니다. 2번째 파라미터는 1번째 실행의 시간 간격이고, 3번째 파라미터는 재실행 시간 간격입니다. 0을 지정하면 AppControl 이벤트가 한번만 실행됩니다. 4번째 파라미터에는 Alarm의 ID가 반환됩니다. Alarm을 종료할 때 필요합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 3초 후에 HelloWorld 예제가 실행됩니다.



3) Timer Alarm 종료

Alarm 예제에 Button을 추가해서 Alarm을 종료하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Button-1 */
Evas_Object *btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Start Timer Alarm");
evas_object_smart_callback_add(btn, "clicked", btn_start_timer_cb, ad);
/* expand both horiz and vert, fill horiz and vert */
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Stop Timer Alarm");
evas_object_smart_callback_add(btn, "clicked", btn_stop_timer_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
```

그리고 `create_base_gui()` 함수 위에 Button 콜백 함수를 생성하고, `btn_start_timer_cb()` 함수를 수정합니다.

```
static void
btn_start_timer_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    int ret;
    int DELAY = 3;
//int REMIND = 0;
int REMIND = 8;
```

```

app_control_h app_control = NULL;
ret = app_control_create(&app_control);
ret = app_control_set_operation(app_control, APP_CONTROL_OPERATION_DEFAULT);
ret = app_control_set_app_id (app_control, "org.example.helloworld");
ret = alarm_schedule_after_delay(app_control, DELAY, REMIND, &ad->timer_id);
dlog_print(DLOG_DEBUG, "tag", "result = %d", ret);
elm_object_text_set(ad->label, "Timer Alarm Started");

}

static void
btn_stop_timer_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    alarm_cancel(ad->timer_id);
    elm_object_text_set(ad->label, "Timer Alarm Stopped");
}

```

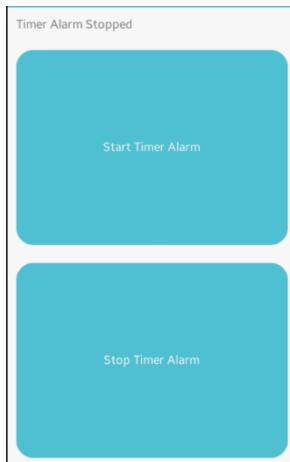
REMIND 변수의 값을 8로 변경하면 HelloWorld 예제가 8초마다 재실행됩니다.

btn_stop_timer_cb()는 사용자가 Button을 눌렀을 때 Timer Alarm을 종료하는 함수입니다.

alarm_cancel(int alarm_id)는 Alarm을 종료하는 API입니다. Alarm의 ID를 전달합니다.

예제를 다시 실행시켜서 1번째 Button을 눌러봅시다. HelloWorld 예제가 실행됩니다. HelloWorld 예제를 종료하면 8초마다 재실행됩니다. 2번째 Button을 누르면 더 이상 HelloWorld 예제가 실행되지 않습니다. Alarm이 종료된 것입니다.

Alarm을 종료하려면 ID가 꼭 필요하기 때문에 Alarm을 시작한 후에는 2번째 Button을 눌러서 Alarm을 종료해야 합니다. 만약 Stop 버튼을 눌러도 HelloWorld 예제가 계속 실행된다면 에뮬레이터를 삭제했다가 다시 생성하면 초기화 됩니다.



3) Date Alarm

특정 시간을 지정해서 원하는 시간에 특정 앱이 실행되는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 2개의 Button 생성 코드를 추가합니다.

```
/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Stop Timer Alarm");
evas_object_smart_callback_add(btn, "clicked", btn_stop_timer_cb, ad);
my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Start Date Alarm");
```

```

    evas_object_smart_callback_add(btn, "clicked", btn_start_date_cb, ad);
    my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

    /* Button-4 */
    btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "Stop Date Alarm");
    evas_object_smart_callback_add(btn, "clicked", btn_stop_date_cb, ad);
    my_box_pack(box, btn, 1.0, 1.0, -1.0, -1.0);

}
}

```

3번째, 4번째 Button을 생성하는 코드입니다. 그런 다음
create_base_gui() 함수 위에 Button 콜백 함수를 생성합니다.

```

static void
btn_start_date_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    int ret;

    struct tm date;
    ret = alarm_get_current_time(&date);
    date.tm_sec+=4;

    app_control_h app_control = NULL;
    ret = app_control_create(&app_control);
    ret = app_control_set_operation(app_control, APP_CONTROL_OPERATION_DEFAULT);
    ret = app_control_set_app_id (app_control, PACKAGE);
    ret = alarm_schedule_at_date(app_control, &date, 0, &ad->date_id);
    elm_object_text_set(ad->label, "Date Alarm Started");
}

static void

```

```
btn_stop_date_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    alarm_cancel(ad->date_id);
    elm_object_text_set(ad->label, "Date Alarm Stopped");
}
```

btn_start_date_cb() 는 Date Alarm을 시작하는 함수입니다.

struct tm 은 날짜와 시간을 저장할 수 있는 시간 구조체 입니다.

alarm_get_current_time(struct tm *) 는 현재 시간을 struct tm 형식으로 반환하는 API입니다.

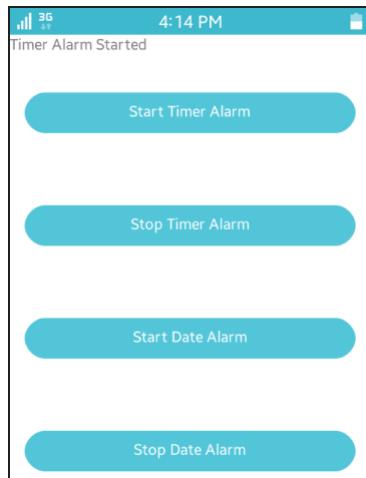
tm.tm_sec에는 초단위 시간이 저장되어 있습니다. 현재로 부터 4초 후를 지정하였습니다.

alarm_schedule_at_date(app_control_h, struct tm *, int, int *) 는 특정 시간에 AppControl 이벤트를 실행하는 API입니다. 2번째 파라미터에는 날짜와 시간이 저장된 struct tm 객체를 전달하고, 3번째 파라미터에는 재실행 시간 간격을 지정합니다. 0을 전달하면 1번만 실행됩니다. 4번째 파라미터에는 Alarm의 ID가 반환됩니다. Alarm을 종료할 때 필요합니다.

btn_stop_date_cb() 은 Data Alarm을 종료하는 함수입니다.

예제를 다시 실행시켜 보겠습니다. 3번째 Button을 누르고 앱을 종료하면 잠시 후에 Alarm 예제가 다시 실행됩니다.

이번에는 3번째 Button을 누르고 곧 바로 4번째 Button을 누릅니다. 그리고 앱을 종료하면 시간이 흘러도 아무런 변화가 없습니다. Date Alarm이 종료된 것입니다.



4) 관련 API

app_control_h : AppControl 구조체.

int app_control_create(app_control_h *) : AppControl 객체를 생성하는 API.

int app_control_set_operation(app_control_h, char *) : AppControl의 역할을 지정하는 API. 외부 앱을 Loading 할 때는 APP_CONTROL_OPERATION_DEFAULT를 전달하면 됩니다.

int app_control_set_app_id(app_control_h, char *) : Loading 할 예제의 패키지명을 AppControl에 지정하는 API. HelloWorld 예제의 패키지명은 "org.example.helloworld"입니다.

int alarm_schedule_after_delay(app_control_h, int, int, int *) : 일정 시간이 경과한 후에 AppControl 이벤트를 전달하는 API. 2번째 파라미터는 1번째 실행의 시간 간격이고, 3번째 파라미터는 재실행 시간 간격입니다. 0을 지정하면 AppControl 이벤트가 한번만 실행됩니다.

4번째 파라미터에는 Alarm의 ID가 반환됩니다. Alarm을 종료할 때 필요합니다.

int alarm_cancel(int alarm_id) : Alarm을 종료하는 API. Alarm의 ID를 전달합니다.

struct tm : 날짜와 시간을 저장할 수 있는 시간 구조체.

int alarm_get_current_time(struct tm *) : 현재 시간을 struct tm 형식으로 반환하는 API.

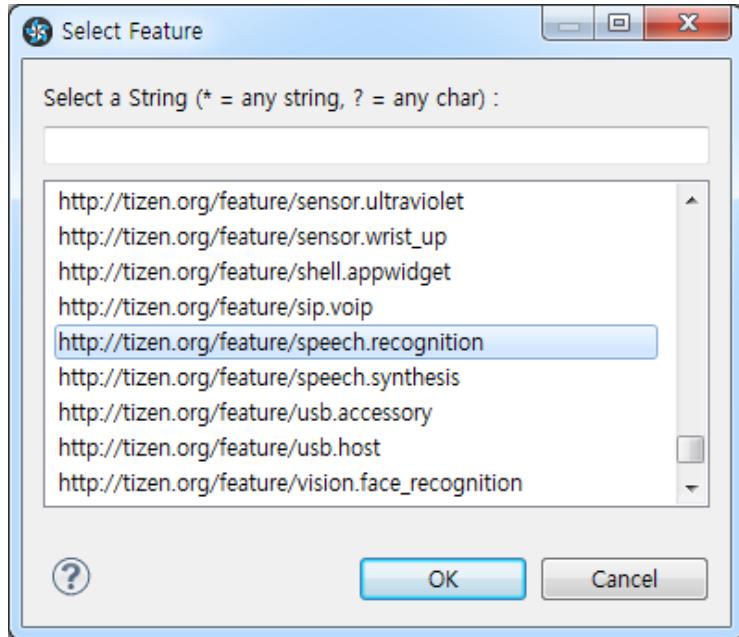
int alarm_schedule_at_date(app_control_h, struct tm *, int, int *) : 특정 시간에 AppControl 이벤트를 실행하는 API. 2번째 파라미터에는 날짜와 시간이 저장된 struct tm 객체를 전달하고, 3번째 파라미터에는 재실행 시간 간격을 지정합니다. 0을 전달하면 1번만 실행됩니다. 4번째 파라미터에는 Alarm의 ID가 반환됩니다. Alarm을 종료할 때 필요합니다.

65. TTS (Text to Speech)

문자를 사람의 목소리로 읽어주는 기능을 TTS(Text to Speech) 라고 하고, 반대로 사람의 목소리를 문자로 인식하는 기능을 STT(Speech to Text) 라고 합니다. 이번 예제에서는 TTS를 사용하는 방법을 알아보겠습니다.

1) Feature 등록

새로운 소스 프로젝트를 생성하고 Project name을 TtsEx 으로 지정합니다. TTS 기능을 사용하기 위해서는 Feature 등록이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Features를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/feature/speech.recognition> 을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



동일한 과정을 반복해서 다음의 Feature를 추가합니다.

- http://tizen.org/feature/speech.synthesis

저장하고 아래쪽 텁버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<manifest xmlns="http://tizen.org/ns/packages" api-version="2.3"
package="org.example.ttsex" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.ttsex" exec="ttsex" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>ttsex</label>
        <icon>ttsex.png</icon>
    </ui-application>
    <feature name="http://tizen.org/feature/speech.recognition">true</feature>
    <feature name="http://tizen.org/feature/speech.synthesis">true</feature>
</manifest>
```

2) TTS 생성 & 삭제

src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "ttsex.h"
#include <tts.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
Evas_Object *entry;
Evas_Object *button;
tts_h tts;
} appdata_s;
```

tts.h는 TTS 를 사용하기 위한 라이브러리 헤더파일 입니다.

entry 위젯에는 문자열을 입력받도록 하겠습니다.

Button 위젯을 클릭하면 문자열 재생을 시작하겠습니다.

tts_h는 TTS 구조체 입니다.

앱이 실행되면 자동으로 TTS 객체를 생성하고, 앱이 종료될 때 TTS 객체를 삭제하는 기능을 구현해 보겠습니다. create_base_gui() 함수 위에

5개의 새로운 함수를 생성합니다.

```
static void
state_changed_cb(tts_h tts, tts_state_e previous, tts_state_e current, void* user_data)
{
    appdata_s *ad = user_data;

    switch (current)
    {
        case TTS_STATE_PLAYING:
            elm_object_text_set(ad->button, "Stop");
            break;
        case TTS_STATE_READY:
        default:
            elm_object_text_set(ad->button, "Play");
            break;
    }
}

static void
utterance_completed_cb(tts_h tts, int utt_id, void *user_data)
{
    appdata_s *ad = user_data;

    dlog_print(DLOG_INFO, LOG_TAG, "Utterance completed: %d", utt_id);
    elm_object_text_set(ad->button, "Stop (idle)");
}

static void
utterance_started_cb(tts_h tts, int utt_id, void *user_data)
{
    appdata_s *ad = user_data;

    dlog_print(DLOG_INFO, LOG_TAG, "Utterance started: %d", utt_id);
```

```

        elm_object_text_set(ad->button, "Stop (speaking)");
    }

static tts_h
create_tts_handle(appdata_s *ad)
{
    tts_h tts;
    int ret = tts_create(&tts);
    if (TTS_ERROR_NONE != ret)
    {
        dlog_print(DLOG_INFO, "tag", "%s err = %d", __func__, ret);
    }
    else
    {
        tts_set_utterance_started_cb(tts, utterance_started_cb, ad);
        tts_set_utterance_completed_cb(tts, utterance_completed_cb, ad);
        tts_set_state_changed_cb(tts, state_changed_cb, ad);
        tts_prepare(tts);
    }

    return tts;
}

static void
destroy_tts_handle(tts_h tts)
{
    int ret = tts_destroy(tts); // tts is the TTS handle
    if (TTS_ERROR_NONE != ret)
    {
        dlog_print(DLOG_INFO, "tag", "%s err = %d", __func__, ret);
    }
}

```

state_changed_cb() 는 TTS 상태 변경 이벤트 함수입니다. 파라미터는

순서대로 TTS 객체, 이전 상태값, 현재 상태값, 사용자 데이터입니다. 기능은 상태값을 문자열로 변경해서 화면에 출력합니다.

tts_state_e 는 TTS 상태 정보를 저장하는 INT 형 변수입니다. 상태 종류는 다음과 같습니다.

- TTS_STATE_CREATED : TTS 생성
- TTS_STATE_READY : 준비 완료
- TTS_STATE_PLAYING : TTS 재생중
- TTS_STATE_PAUSED : 일시정지

utterance_completed_cb() 는 TTS 재생 완료 이벤트 함수입니다.

utterance_started_cb() 는 TTS 재생 시작 이벤트 함수입니다.

create_tts_handle() 는 TTS 객체를 생성해서 반환하는 함수입니다.

tts_create(tts_h*) 는 TTS 객체를 생성하는 API 입니다.

tts_set_utterance_started_cb() 는 TTS 재생 시작 이벤트 콜백 함수명을 지정하는 API 입니다.

tts_set_utterance_completed_cb() 는 TTS 재생 완료 이벤트 콜백 함수명을 지정하는 API 입니다.

tts_set_state_changed_cb(tts_h, tts_state_changed_cb, void*) 는 TTS 상태 변경 이벤트 콜백 함수명을 지정하는 API입니다. 파라미터는 순서대로 TTS 객체, 콜백 함수명, 사용자 데이터 입니다.

tts_prepare(tts_h) 는 TTS 객체를 준비 상태로 전환하는 API 입니다.

destroy_tts_handle() 는 TTS 객체를 삭제하는 함수입니다.

`tts_destroy(tts_h)` 는 TTS 객체를 삭제하는 API입니다.

위 함수들을 앱 구동될 때와 종료될 때에 호출하면 됩니다.

`create_base_gui()` 함수 위에 2개의 새로운 함수를 생성합니다.

```
static void
btn_play_cb(void *data, Evas_Object *obj, void *event_info)
{
}

static void
my_box_pack(Evas_Object *box, Evas_Object *child, double h_weight, double v_weight,
            double h_align, double v_align)
{
    /* create a frame we shall use as padding around the child widget */
    Evas_Object *frame = elm_frame_add(box);
    /* use the medium padding style. there is "pad_small", "pad_medium",
     * "pad_large" and "pad_huge" available as styles in addition to the
     * "default" frame style */
    elm_object_style_set(frame, "pad_medium");
    /* set the input weight/aling on the frame insted of the child */
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);
    evas_object_size_hint_align_set(frame, h_align, v_align);
    {
        /* tell the child that is packed into the frame to be able to expand */
        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
                                         EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposaed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL,
                                         EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
```

```

        elm_object_content_set(frame, child);
    }

/* put the frame into the box instead of the child directly */
elm_box_pack_end(box, frame);

/* show the frame */
evas_object_show(frame);

}

```

btn_play_cb() 는 Button 이벤트 함수입니다. 함수 내용은 잠시 후에 입력해 보겠습니다.

my_box_pack() 은 Box에 위젯을 추가하는 함수입니다.

create_base_gui() 함수에 새로운 코드를 추가합니다. Box, Entry, Button을 생성하는 코드입니다. 그리고 TTS 객체를 생성하는 함수를 호출합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

{
    Evas_Object *btn, *box;

    /* Container: standard box */
    box = elm_box_add(ad->win);
    elm_box_homogeneous_set(box, EINA_TRUE);
    elm_box_horizontal_set(box, EINA_FALSE);

```

```

evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
evas_object_size_hint_align_set(box, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* Entry */
    ad->entry = elm_entry_add(box);
    elm_entry_single_line_set(ad->entry, EINA_FALSE);
    elm_entry_scrollable_set(ad->entry, EINA_TRUE);
    elm_object_text_set(ad->entry, "Hello world");
    my_box_pack(box, ad->entry, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND,
EVAS_HINT_FILL, EVAS_HINT_FILL);

    /* Button-1 */
    btn = elm_button_add(box);
    elm_object_text_set(btn, "Play/Stop");
    evas_object_smart_callback_add(btn, "clicked", btn_play_cb, ad);
    my_box_pack(box, btn, EVAS_HINT_EXPAND, EVAS_HINT_EXPAND,
EVAS_HINT_FILL, 0.0);
    ad->button = btn;
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

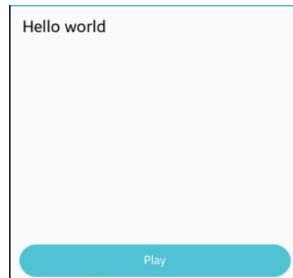
ad->tts = create_tts_handle(ad);
}

```

그리고 app_terminate() 함수에도 새로운 코드를 추가합니다. 앱이 종료될 때 TTS 객체도 삭제됩니다.

```
static void  
app_terminate(void *data)  
{  
    appdata_s *ad = data;  
    destroy_tts_handle(ad->tts);  
}
```

예제를 빌드하고 실행시켜 봅시다. TTS 객체가 생성되고 Ready 상태가 되면 Button 캡쳐 텍스트가 Play로 변경됩니다. 이제 사용할 준비가 된 것입니다.



3) 문자열을 음성으로 변환

Entry 위젯에 문자열을 입력하고 Button을 누르면 문자열이 음성으로 출력되는 기능을 구현해 보겠습니다.

create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다. 그리고 Button 이벤트 함수에 내용을 입력합니다.

```
static void  
add_text(tts_h tts, appdata_s *ad)  
{
```

```

const char* text = "Good morning"; // Text for read
const char* language = "en_US"; // Language
int voice_type = TTS_VOICE_TYPE_FEMALE; // Voice type
int speed = TTS_SPEED_AUTO;
int utt_id; // Utterance ID for the requested text
text = elm_object_text_get(ad->entry);

int ret = tts_add_text(tts, text, language, voice_type, speed, &utt_id);
if (TTS_ERROR_NONE != ret)
{
    dlog_print(DLOG_INFO, "tag", "%s err = %d", __func__, ret);
}
}

static int
get_state(tts_h* tts)
{
    tts_state_e current_state;
    int ret;
    ret = tts_get_state(*tts, &current_state);

    if (TTS_ERROR_NONE != ret)
    {
        dlog_print(DLOG_INFO, "tag", "%s state = %d", __func__, ret);
        return -1;
    }
    else
    {
        dlog_print(DLOG_INFO, "tag", "%s state = %d", __func__, current_state);
        return (int) current_state;
    }
}

static void
btn_play_cb(void *data, Evas_Object *obj, void *event_info)

```

```

{
    appdata_s *ad = data;
    int state = get_state(&ad->tts);
    if ((tts_state_e) state == TTS_STATE_READY || (tts_state_e) state ==
TTS_STATE_PAUSED)
    {
        add_text(ad->tts, ad);
        int ret = tts_play(ad->tts);
        if (TTS_ERROR_NONE != ret)
        {
            dlog_print(DLOG_INFO, "tag", "%s err = %d", __func__, ret);
        }
    }
    else if ((tts_state_e) state == TTS_STATE_PLAYING)
    {
        int ret = tts_stop(ad->tts);
        if (TTS_ERROR_NONE != ret)
        {
            dlog_print(DLOG_INFO, "tag", "%s err = %d", __func__, ret);
        }
    }
}

```

`add_text()` 는 TTS 객체에 문자열을 추가하는 함수입니다.

`tts_add_text(tts_h, char*,char*, int, int, int*)` 는 TTS 객체에 문자열을 추가하는 API입니다. 파라미터는 순서대로 TTS 객체, 문자열, 목소리 종류, 속도입니다.

목소리 종류는 다음과 같습니다.

- `TTS_VOICE_TYPE_AUTO` : 목소리 종류 자동
- `TTS_VOICE_TYPE_MALE` : 남자 목소리

- TTS_VOICE_TYPE_FEMALE : 여자 목소리
- TTS_VOICE_TYPE_CHILD : 어린이 목소리

get_state() 는 TTS의 현재 상태를 반환하는 함수입니다.

tts_get_state(tts_h tts, tts_state_e* state) 는 TTS의 현재 상태를 반환하는 API 입니다.

btn_play_cb() 는 Button을 눌렀을 때 TTS를 시작하는 콜백 함수입니다.
현재 상태가 준비완료 혹은 일시정지 일때는 재생을 시작하고, 재생중 상태일 때는 중지합니다.

tts_play(tts_h tts) 는 TTS 재생을 시작하는 API입니다.

tts_stop(tts_h tts) 는 TTS 재생을 중지하는 API입니다.

예제를 다시 실행시켜 봅시다. Button을 누르면 TTS 재생이 시작됩니다.
한번 더 누르면 재생이 중지되고, 다시 한번 누르면 재시작 합니다.



4) 관련 API

int tts_create(tts_h*) : TTS 객체를 생성하는 API.

int tts_set_state_changed_cb(tts_h tts, tts_state_changed_cb callback,

`void* user_data)` : TTS 상태 변경 이벤트 콜백 함수명을 지정하는 API.
파라미터는 순서대로 TTS 객체, 콜백 함수명, 사용자 데이터 입니다.

`int tts_prepare(tts_h tts)` : TTS 객체를 준비 상태로 전환하는 API.

`int tts_destroy(tts_h tts)` : TTS 객체를 삭제하는 API.

`int tts_add_text(tts_h tts, const char* text, const char* language, int voice_type, int speed, int* utt_id)` : TTS 객체에 문자열을 추가하는 API.
파라미터는 순서대로 TTS 객체, 문자열, 목소리 종류, 속도입니다. 목소리 종류는 다음과 같습니다.

- `TTS_VOICE_TYPE_AUTO` : 목소리 종류 자동
- `TTS_VOICE_TYPE_MALE` : 남자 목소리
- `TTS_VOICE_TYPE_FEMALE` : 여자 목소리
- `TTS_VOICE_TYPE_CHILD` : 어린이 목소리

`int tts_get_state(tts_h tts, tts_state_e* state)` : TTS의 현재 상태를 반환하는 API.

`int tts_play(tts_h tts)` : TTS 재생을 시작하는 API.

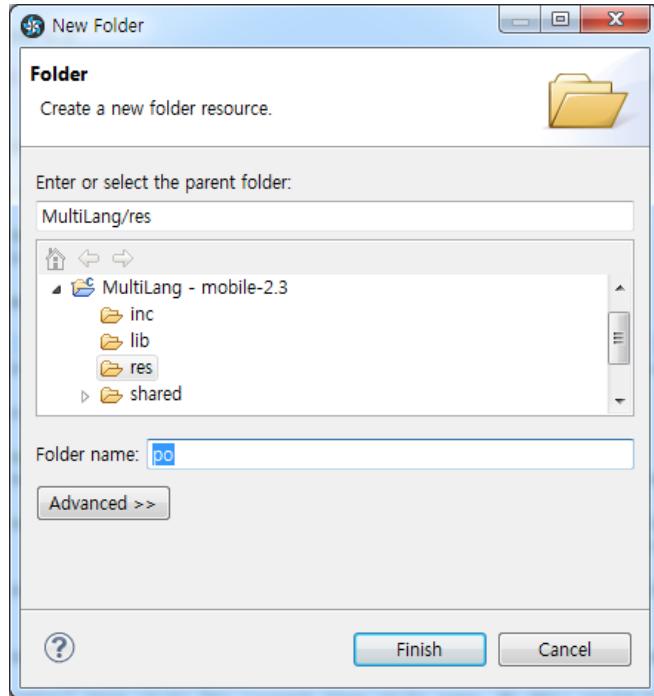
`int tts_stop(tts_h tts)` : TTS 재생을 중지하는 API.

66. 다국어 지원

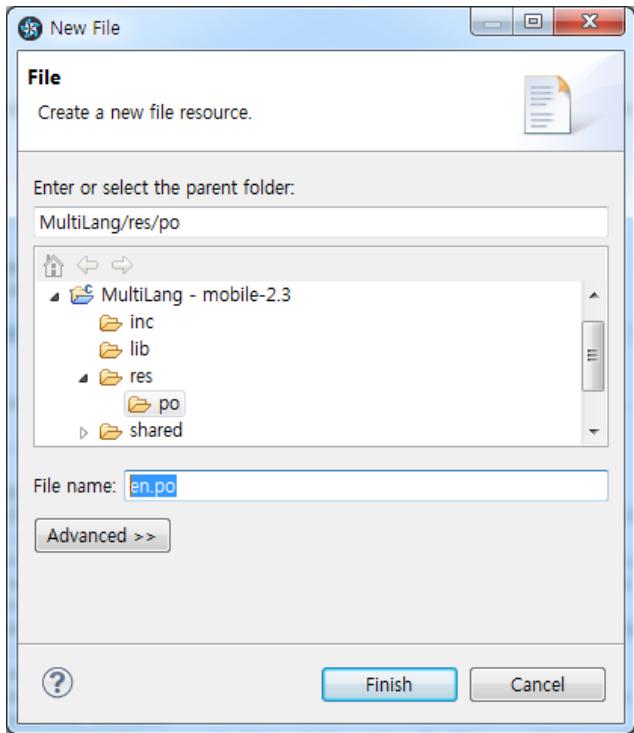
앱 스토어가 개인 개발자에게 개방되면서 아시아에 있는 개발자가 유럽 소비자에게도 앱을 판매할 수 있게 되었습니다. 하지만 여러 가지 언어를 지원하기 위해서 각 언어별로 버전을 다르게 만들어야 한다면 개발자로서는 매우 불편할 것입니다. 동일한 의미를 가지는 각 언어별 텍스트를 리소스에 등록해 놓고 필요한 부분에 호출해서 사용하면 이 문제를 해결할 수 있습니다. 이번 시간에는 PO 파일에 다국어 정보를 저장하고 소스코드에서 호출하는 방법을 알아보겠습니다.

1) 다국어 정보를 리소스에 등록

새로운 소스 프로젝트를 생성하고 Project name을 MultiLang으로 지정합니다. 소스 프로젝트가 생성되었으면 다국어 정보를 등록해 보겠습니다. 다국어 정보는 /res/po 폴더에 저장합니다. /res 폴더를 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [New > Folder]를 선택합니다. 팝업창이 나타나면 Folder name 항목에 po라고 입력하고 Finish 버튼을 누릅니다.

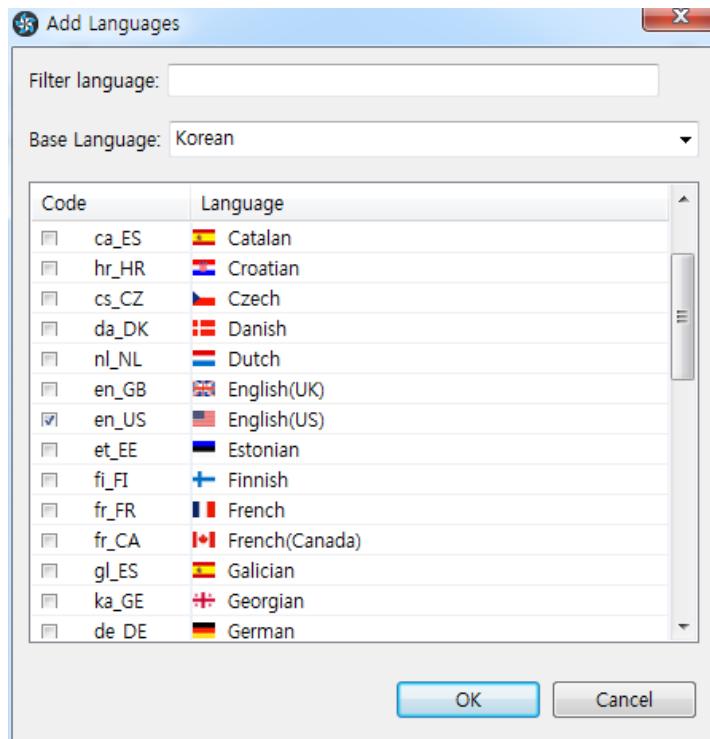


다국어 정보 파일은 자동으로 생성되지는 않습니다. 수동으로 파일을 생성하겠습니다. /res/po 폴더를 마우스 오른쪽 버튼으로 클릭하고 단축메뉴에서 [New > File]을 선택합니다. 팝업창이 나타나면 File name 항목에 en.po이라고 입력하고 Finish 버튼을 누릅니다.



en.po 는 영어 관련 정보를 저장하는 파일입니다. 대부분의 앱은 영어를 지원하기 때문에 거의 필수적인 다국어 정보 파일입니다. 지원하는 언어 개수가 늘어날 때마다 po파일 개수도 늘어납니다. po 파일이 생성되면 에디터에 내용이 표시됩니다. 만약 자동으로 에디터가 열리지 않으면 en.po 파일을 더블 클릭하면 됩니다.

'Add Languages' 팝업창이 나타나면 언어 목록에서 몇가지를(en_US(English), ko_KR(Korean), ru_RU(Russian)) 체크 하겠습니다.



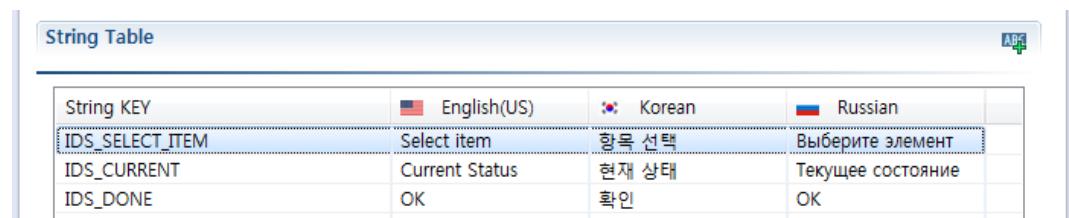
메시지를 하나 추가해 보겠습니다. 화면 우측에 'Add String Key' 버튼을 누르고 MsgID에 IDS_DONE이라고 입력합니다. 그런 다음 English 컬럼에 해당하는 필드에 OK라고 입력합니다. Korean 필드에는 확인, Russian 필드에는 OK를 입력합니다.

The screenshot shows the 'MultiLang Localization' application window. The title bar says 'MultiLang Localization'. The main area is titled 'String Localization' under a 'Supporting Languages' section. It includes checkboxes for 'Select All', 'English(US)', 'Korean', and 'Russian'. Below this is a 'String Table' section with a table:

String KEY	English(US)	Korean	Russian
IDS_DONE	OK	확인	OK

메시지를 하나 더 추가하겠습니다. 'Add String Key' 버튼을 누르고 새로 추가된 항목에 내용을 입력합니다. (IDS_CURRENT, Current Status, 현재 상태, Текущее состояние)

'Add String Key' 버튼을 한번 더 누르고 새로 추가된 항목에 내용을 입력합니다. (IDS_SELECT_ITEM, Select item, 항목 선택, Выберите элемент)



String KEY	English(US)	Korean	Russian
IDS_SELECT_ITEM	Select item	항목 선택	Выберите элемент
IDS_CURRENT	Current Status	현재 상태	Текущее состояние
IDS_DONE	OK	확인	OK

2) 소스코드에서 다국어 정보 구하기

po 파일을 저장한 다음, src 폴더에 소스파일(~.c)을 열고 appdata 구조체에 변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
Evas_Object* naviframe;
} appdata_s;
```

Label 위젯에 IDS_CURRENT에 해당하는 다국어를 적용시켜 보겠습니다. create_base_gui() 함수로 이동해서 Label 위젯 생성 코드를 수정합니다.

```

/* Label*/
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, i18n_get_text("IDS_CURRENT"));
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_object_content_set(ad->conform, ad->label);

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

`i18n_get_text(const char *message)` 는 환경설정의 언어에 해당하는
다국어 정보를 반환하는 API입니다.

예제를 빌드하고 실행시켜 봅시다. po 파일에 등록했던 다국어 정보가
표시됩니다.



3) NaviFrame에 다국어 적용

헤더의 제목 텍스트에 다국어를 적용하려면 `i18n_get_text()` 함수를
사용하면 됩니다. 환경설정 언어 변경시 자동으로 적용되려면 다른
방법을 사용해야 합니다. `create_base_gui()` 함수에 NaviFrame 과 Box
생성 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);

```

```

elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Naviframe */
ad->naviframe = elm_naviframe_add(ad->conform);
eext_object_event_callback_add(ad->naviframe, EEXT_CALLBACK_BACK,
win_back_cb, ad);
elm_object_content_set(ad->conform, ad->naviframe);

/* Box */
Evas_Object *box = elm_box_add(ad->naviframe);
elm_box_padding_set(box, 0, ELM_SCALE_SIZE(20));

/* Push a view to naviframe */
Elm_Object_Item *nf_it = elm_naviframe_item_push(ad->naviframe,
"IDS_SELECT_ITEM", NULL, NULL, box, NULL);
/* Mark naviframe title as translatable text */
elm_object_item_part_text_translatable_set(nf_it, NULL, EINA_TRUE);

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, i18n_get_text("IDS_CURRENT"));
    //evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    //elm_object_content_set(ad->conform, ad->label);
    evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND, 0.0);
    evas_object_size_hint_align_set(ad->label, 0.5, 0.0);
    elm_box_pack_end(box, ad->label);
    evas_object_show(ad->label);
}

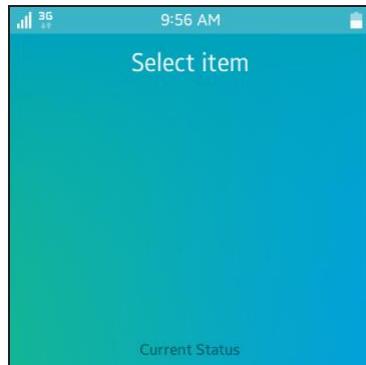
```

```
/* Show window after base gui is set up */  
evas_object_show(ad->win);
```

Naviframe 과 Box 컨테이너를 생성하고 제목 텍스트를 "IDS_SELECT_ITEM" 으로 지정했습니다. 이렇게 하면 제목이 "IDS_SELECT_ITEM" 으로 표시됩니다. 그래서 뭔가 추가해줄 필요가 있습니다.

elm_object_item_part_text_translatable_set(it, part, translatable) 는 객체의 텍스트를 리소스에서 구하도록 지정하는 API입니다. 3번째 파라미터가 EINA_TRUE 이면 리소스에서 텍스트를 가져옵니다. EINA_FALSE 이면 입력된 텍스트를 그대로 사용합니다.

예제를 다시 실행시켜 봅시다. 헤더가 표시되고 제목에 다국어가 적용되었습니다.



4) 위젯에 다국어 적용

Label, Button, Entry 같은 위젯에 다국어를 적용하는 방법은 따로 있습니다. create_base_gui() 함수에 Button 과 Label 생성 코드를 추가합니다.

```

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, i18n_get_text("IDS_CURRENT"));
    evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND, 0.0);
    evas_object_size_hint_align_set(ad->label, 0.5, 0.0);
    elm_box_pack_end(box, ad->label);
    evas_object_show(ad->label);

    /* Button */
    Evas_Object* btn = elm_button_add(ad->conform);
    elm_object_translatable_text_set(btn, "IDS_DONE");
    evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0.0);
    evas_object_size_hint_align_set(btn, -1.0, 0.0);
    elm_box_pack_end(box, btn);
    evas_object_show(btn);

    /* Label*/
    Evas_Object *o = elm_label_add(ad->conform);
    elm_label_line_wrap_set(o, ELM_WRAP_WORD);
    elm_object_text_set(o, "This label is not translatable.<br/>"
        "Go to Settings to change the device language and see how the above
items will get translated.<br/>"
        "Supported languages are: English, Korean, Russian.");
    evas_object_size_hint_weight_set(o, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(o, -1.0, 0.0);
    elm_box_pack_end(box, o);
    evas_object_show(o);
}

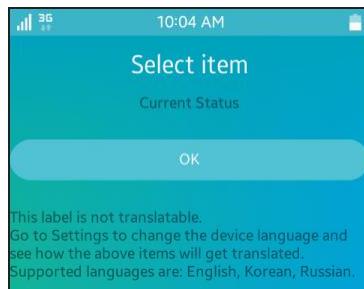
```

elm_object_translatable_text_set(obj, text) 는 위젯에 리소스로 등록된

텍스트를 자동으로 적용시켜 주는 API입니다. 사용자가 환경설정에서 언어를 변경하면 자동으로 위젯의 텍스트에도 다국어가 재적용됩니다.

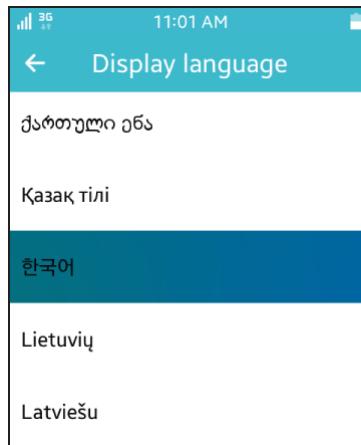
2번째 Label 위젯은 사용자에게 텍스트 문구를 표시하기 위해서 사용되었으며 특별한 기능은 없습니다.

예제를 다시 실행시켜 봅시다. Button 위젯에 다국어가 적용되었습니다.

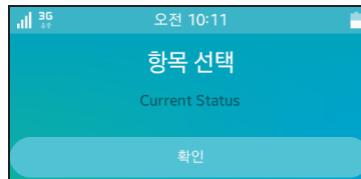


5) 다국어 자동 변경

에뮬레이터의 기본 언어 설정은 English(United States)입니다. 다른 언어로 변경해 보겠습니다. MultiLang 앱이 실행된 상태에서 Back 버튼을 누르고 앱목록 화면으로 이동합니다. Settings 아이콘을 누르고 목록 중에서 'Language and input'을 선택합니다. 화면이 바뀌면 'Display language'를 선택합니다. 언어 목록이 나타나면 한국어, 러시아어 중에서 한가지를 선택합니다.



환경설정을 빠져나와서 MultiLang 예제를 다시 실행시켜 봅시다. Home 버튼을 오래 누르면 이제까지 실행했던 앱 목록이 나타나고 그중에서 MultiLang을 선택하면 됩니다. 예제가 Foreground 모드로 전환되면 NaviFrame과 Button에는 자동으로 다국어가 변경되었습니다. Label 위젯에는 영어에서 변경되지 않았습니다. 환경설정 언어가 변경되는 이벤트를 구해서 수동으로 다국어를 재지정해야 합니다.



소스파일 아래쪽으로 이동하면 ui_app_lang_changed() 이라는 함수가 있습니다. 이 함수 끝부분에 새로운 코드를 추가합니다.

```
static void
ui_app_lang_changed(app_event_info_h event_info, void *user_data)
{
    /*APP_EVENT_LANGUAGE_CHANGED*/
    char *locale = NULL;
```

```

    system_settings_get_value_string(SYSTEM_SETTINGS_KEY_LOCALE_LANGUAGE,
&locale);
    elm_language_set(locale);
    free(locale);

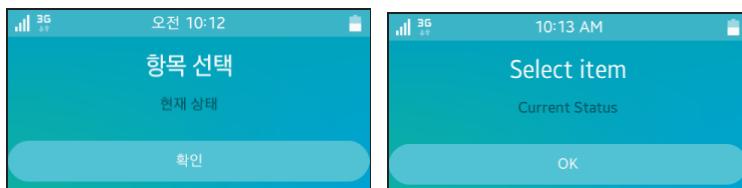
appdata_s* ad = user_data;
elm_object_text_set(ad->label, i18n_get_text("IDS_CURRENT"));
}

```

ui_app_lang_changed() 는 환경설정의 언어 종류가 변경될 때 실행되는 이벤트 함수입니다. 함수명을 변경하려면 main() 함수에서 수정하면 됩니다.

label 위젯에 다국어를 재지정 하였습니다. 이렇게 하면 사용자가 환경설정에서 언어 종류를 변경할 때 자동으로 위젯에 다국어 설정이 반영됩니다.

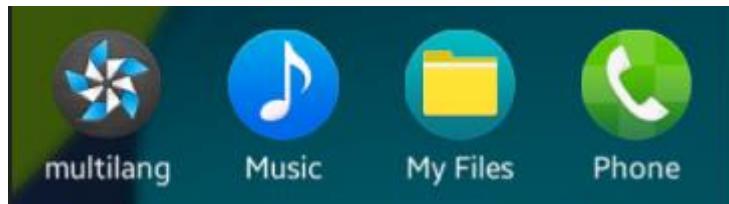
예제를 다시 실행하고 환경설정에서 언어를 English(United States)로 변경해 봅시다. 다시 MultiLang 예제로 돌아오면 자동으로 다국어가 적용됩니다.



6) App 아이콘 텍스트 변경

Back 버튼을 누르고 앱 아이콘 목록 화면으로 이동하면 MultiLang 예제의 아이콘이 보입니다. Label 텍스트는 multilang 으로 표시되어

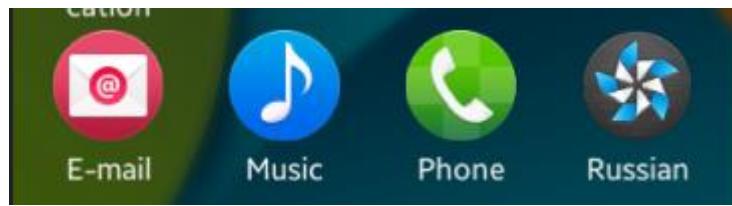
있을 것입니다. 여기에 다국어를 적용하는 방법을 알아보겠습니다.



tizen-manifest.xml 파일을 열고 아래쪽 텁버튼 중에서 tizen-manifest.xml 버튼을 누르면 소스코드가 보입니다. <label>multilang</label> 라는 코드가 보이는데 이것이 앱 아이콘의 Label 텍스트입니다. 새로운 XML 코드를 추가합니다.

```
<manifest xmlns="http://tizen.org/ns/packages" api-version="2.3"
package="org.example.multilang" version="1.0.0">
    <profile name="mobile" />
    <ui-application appid="org.example.multilang" exec="multilang" type="capp"
multiple="false" taskmanage="true" nodisplay="false">
        <icon>multilang.png</icon>
        <label>multilang</label>
        <label xml:lang="en-us">English</label>
        <label xml:lang="ko-kr">Korean</label>
        <label xml:lang="ru-ru">Russian</label>
    </ui-application>
</manifest>
```

label 태그에 언어별로 텍스트를 지정하면 다국어가 적용됩니다. 예제를 다시 실행하고 환경설정에서 언어를 한국어 또는 러시아어로 변경해 봅시다. 그리고 앱 아이콘 목록 화면으로 돌아오면 앱 아이콘 텍스트가 변경됩니다.



7) 관련 API

`char* i18n_get_text(const char *message)` : 환경설정의 언어에 해당하는
다국어 정보를 반환하는 API.

`void elm_object_translatable_text_set(obj, text)` : 위젯에 리소스로 등록된
텍스트를 자동으로 적용시켜 주는 API. 사용자가 환경설정에서 언어를
변경하면 자동으로 위젯의 텍스트에도 다국어가 재적용됩니다.

`void elm_object_item_part_text_translatable_set(it, part,
translatable)` : 객체의 텍스트를 리소스에서 구하도록 지정하는 API.
3번째 파라미터가 EINA_TRUE 이면 리소스에서 텍스트를 가져옵니다.
EINA_FALSE 이면 기본 텍스트를 그대로 사용합니다.

`void ui_app_lang_changed()` : 환경설정의 언어 종류가 변경될 때
실행되는 이벤트 함수. 함수명을 변경하려면 main() 함수에서 수정하면
됩니다.

67. JSON 파싱

JSON (JavaScript Object Notation) 은 XML 과 마찬가지로 데이터 전송에 사용되는 규약입니다. 이름에서 알 수 있듯이 자바스크립트에서 주로 사용되었으나 XML에 비해서 사용이 간편해서 현재는 웹기반 통신에 널리 사용되고 있습니다. 예를 들어 아래 XML 구문과 JSON 구문은 동일한 내용이지만 JSON 쪽이 훨씬 간편한 것을 알 수 있습니다. 꼭 필요한 정보만 포함하기 때문입니다. 이번 예제에서는 JSON 데이터를 파싱하는 방법을 알아보겠습니다.

XML 형식 - <name>Obama</name>50

JSON 형식 - [name:Obama, math:50]

1) JSON 배열 데이터 구하기

새로운 소스 프로젝트를 생성하고 Project name을 JsonParse 으로 지정합니다. src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일을 추가합니다.

```
#include "jsonparse.h"  
#include <json-glib/json-glib.h>
```

json-glib/json-glib.h 는 JSON 파싱을 위한 라이브러리 헤더파일입니다.

JSON 형식에서는 '[]' 기호 안에 배열을 저장합니다. 배열 안의 데이터를 하나씩 추출하는 방법을 알아보겠습니다. create_base_gui() 함수 위에

새로운 함수를 생성합니다.

```
static void  
parse_json(appdata_s *ad)  
{  
    JsonParser *parser = json_parser_new ();  
    char buf[256];  
    buf[0] = 'W0';  
  
    const char* data1 = "[11, 22, 33, 44, 55]";  
  
    if( json_parser_load_from_data( parser, data1, strlen(data1), NULL))  
    {  
        JsonNode *root = json_parser_get_root (parser);  
        JSONArray *temp_array = json_node_get_array ( root );  
        for(int i=0; i < json_array_get_length(temp_array); i++ )  
            sprintf(buf, "%s - %d", buf, json_array_get_int_element(temp_array, i));  
    }  
  
    elm_object_text_set(ad->label, buf);  
}
```

JsonParser 는 JSON 데이터 파싱에 사용되는 구조체 입니다.

json_parser_new() 는 JsonParser 객체를 생성하는 API 입니다.

data1이라는 이름의 문자열 변수에 JSON 구문을 저장하였습니다. 5개의 숫자 항목을 가진 배열입니다.

json_parser_load_from_data(JsonParser*, gchar*, gssize, GError**) 는 JsonParser 객체에 JSON 구문을 입력하는 API입니다. 파라미터는 순서대로 JsonParser 객체, JSON 데이터, 데이터 길이, 에러 코드 입니다.

`json_parser_get_root(JsonParser*)` 는 JsonParser의 시작 위치를 반환하는 API입니다. 반환 형식은 JsonNode입니다.

`json_node_get_array(JsonNode*)` 는 Json 구문을 배열 형식으로 반환하는 API입니다. 반환 형식은 JSONArray입니다.

`json_array_get_length(JsonArray*)` 는 Json 배열의 항목 개수를 반환하는 API입니다. 반환 형식은 guint입니다.

`json_array_get_int_element(JsonArray*, guint)` 는 Json 배열의 특정 항목을 정수형으로 변환해서 반환하는 API입니다. 반환 형식은 gint64입니다.

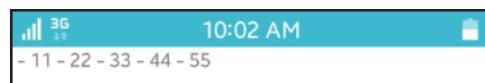
예제가 실행되면 자동으로 위 함수가 실행하겠습니다. `create_base_gui()` 함수 끝부분에 한줄의 코드를 추가합니다.

```
/* Show window after base gui is set up */
evas_object_show(ad->win);

parse_json(ad);

}
```

예제를 빌드하고 실행시켜 보겠습니다. Json 배열의 저장된 5개의 데이터가 Label 위젯에 출력되었습니다.



2) Key로 데이터 구하기

Json 구문은 일반적으로 다음과 같은 형식을 주로 사용합니다. name이 Key이고, Obama가 데이터입니다.

```
{name:Obama, math:50}
```

Key 값으로 데이터를 구하는 방법을 알아보겠습니다. parse_json() 함수에 새로운 코드를 추가합니다.

```
~  
const char* data2 = "{time: '03:53:25 AM', 'millisec_epoch': 1362196405309,  
'date': '03-02-2013'}";  
if( json_parser_load_from_data( parser, data2, strlen(data2), NULL))  
{  
    JsonNode *root = json_parser_get_root (parser);  
    JsonObject *obj = json_node_get_object(root);  
    char* time_data = json_object_get_string_member (obj, "time");  
    long long epoch_data = json_object_get_int_member (obj, "millisec_epoch");  
    sprintf(buf, "%s <br/><br/> time - %s <br/> epoch - %lld", buf, time_data,  
epoch_data);  
}  
  
elm_object_text_set(ad->label, buf);  
}
```

data2라는 이름의 문자열 변수에 Json 구문을 저장하였고, json_parser_load_from_data() 함수를 사용해서 JsonParser 객체에 Json 구문을 입력하였습니다.

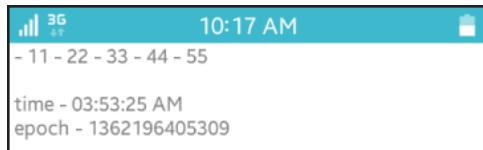
json_node_get_object(JsonNode*)는 Json 구문을 JsonObject 형식으로

반환하는 API 입니다.

`json_object_get_string_member(JsonObject*, gchar*)` 는 JsonObject에서 데이터를 문자열 형식으로 반환하는 API입니다. 2번째 파라미터에 Key를 전달하면 `gchar*` 형식으로 반환합니다.

`json_object_get_int_member(JsonObject*, gchar*)` 는 JsonObject에서 데이터를 숫자 형식으로 반환하는 API입니다. 2번째 파라미터에 Key를 전달하면 `gint64` 형식으로 반환합니다.

예제를 다시 실행시켜 봅시다. time 항목 데이터와 epoch 항목 데이터가 화면에 출력되었습니다.



3) 다단계 파싱

다음 Json 구문을 보면 coord 안에 Json 오브젝트와 Json 배열이 포함되어 있습니다. 이번에는 이렇게 복잡한 구조의 Json 구문을 파싱하는 방법을 알아보겠습니다.

```
{'coord':{'lon':127.03, 'lat':37.5}, 'weather':[{'id':800, 'main':'Clear'}, {'id':887, 'main':'Cloudy'}]}
```

`parse_json()` 함수 끝부분에 새로운 코드를 추가합니다.

Json 오브젝트 안에 포함된 Json 오브젝트에서 데이터를 구하는 코드입니다.

```

~  

const char* data3 = "{<coord>:{<lon>:127.03, <lat>:37.5},  

<weather>:[{<id>:800, <main>'Clear'}, {<id>:887, <main>'Cloudy'}]}";  

if( json_parser_load_from_data( parser, data3, strlen(data3), NULL))  

{  

    JsonNode *root = json_parser_get_root (parser);  

    JsonObject *obj = json_node_get_object(root);  

    JsonNode *temp_node = json_object_get_member (obj, "coord");  

    JsonObject *temp_object = json_node_get_object(temp_node);  

    sprintf(buf, "%s <br/><br/> coord:lon - %0.2f", buf,  

           json_object_get_double_member (temp_object, "lon"));  

}  

elm_object_text_set(ad->label, buf);

```

data3라는 이름의 문자열 변수에 Json 구문을 저장하였고, json_parser_load_from_data() 함수를 사용해서 JsonParser 객체에 Json 구문을 입력하였습니다.

json_parser_get_root() 함수로 시작 위치의 Json 노드를 구했고, json_node_get_object() 함수로 Json 구문을 JsonObject 형식으로 변환합니다.

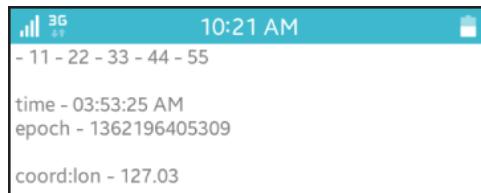
json_object_get_member(JsonObject *, gchar*)는 특정 JsonObject를 반환하는 API입니다. 파라미터는 순서대로 JsonParser, 멤버 이름입니다. 반환 형식은 JsonNode입니다.

json_node_get_object() 함수를 사용해서 JsonNode에서 JsonObject를 구했습니다.

json_object_get_double_member(JsonObject *, gchar*)는

JsonObject에서 데이터를 실수 형식으로 반환하는 API입니다. 2번째 파라미터에 Key를 전달하면 gdouble 형식으로 반환합니다.

예제를 다시 실행시켜 봅시다. coord 그룹 안에 lon Key에 해당하는 데이터가 출력되었습니다.



4) 배열 내부 그룹 데이터 파싱

이번에는 동일한 Json 구문에서 weather라는 이름의 배열의 1번째 항목에서 id에 해당하는 값을 구해 보겠습니다. parse_json() 함수에 새로운 코드를 추가합니다.

```
const char* data3 = "{"coord":{"lon":127.03, "lat":37.5}, "weather":[{"id":800, "main":'Clear'}, {"id":887, "main":'Cloudy'}]}";
if( json_parser_load_from_data( parser, data3, strlen(data3), NULL))
{
    JsonNode *root = json_parser_get_root (parser);
    JsonObject *obj = json_node_get_object(root);
    JsonNode *temp_node = json_object_get_member (obj, "coord");
    JsonObject *temp_object = json_node_get_object(temp_node);
    sprintf(buf, "%s <br/><br/> coord:lon - %0.2f", buf,
            json_object_get_double_member (temp_object, "lon"));

    temp_node = json_object_get_member ( obj,"weather" );
    JSONArray *temp_array = json_node_get_array ( temp_node );
    temp_node = json_array_get_element(temp_array, 0 );
}
```

```

temp_object = json_node_get_object( temp_node );
sprintf(buf, "%s <br/> weather:id - %d", buf, json_object_get_int_member
(temp_object, "id"));

}

elm_object_text_set(ad->label, buf);

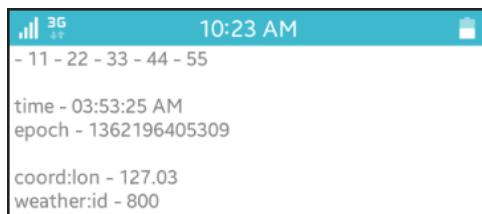
```

json_object_get_member() 함수를 사용해서 weather라는 이름의 배열 노드를 구하고, json_node_get_array() 함수를 사용해서 노드를 JSONArray 형식으로 변환합니다.

json_array_get_element() 함수를 사용해서 1번째 항목 노드를 구하고, json_node_get_object() 함수를 사용해서 노드를 JSONObject 형식으로 변환합니다.

마지막으로 json_object_get_int_member() 함수를 사용해서 id라는 Key에 해당하는 데이터를 정수형으로 변환합니다.

예제를 다시 실행시켜 봅시다. 1번째 배열 항목의 id에 해당하는 값이 출력되었습니다.



5) 관련 API

JsonParser *json_parser_new() : JsonParser 객체를 생성하는 API.

gboolean json_parser_load_from_data(JsonParser*, gchar*, gssize, GError**) : JsonParser 객체에 JSON 구문을 입력하는 API. 파라미터는 순서대로 JsonParser 객체, JSON 데이터, 데이터 길이, 에러 코드입니다.

JsonNode* json_parser_get_root(JsonParser*) : JsonParser의 시작 위치를 반환하는 API. 반환 형식은 JsonNode입니다.

JSONArray* json_node_get_array(JsonNode*) : Json 구문을 배열 형식으로 반환하는 API. 반환 형식은 JSONArray입니다.

quint json_array_get_length(JsonArray*) : Json 배열의 항목 개수를 반환하는 API. 반환 형식은 quint입니다.

gint64 json_array_get_int_element(JsonArray*, guint) : Json 배열의 특정 항목을 정수형으로 변환해서 반환하는 API. 반환 형식은 gint64입니다.

JsonObject* json_node_get_object(JsonNode*) : Json 구문을 JsonObject 형식으로 반환하는 API.

gchar* json_object_get_string_member(JsonObject*, gchar*) : JsonObject에서 데이터를 문자열 형식으로 반환하는 API. 2번째 파라미터에 Key를 전달하면 gchar* 형식으로 반환합니다.

gint64 json_object_get_int_member(JsonObject*, gchar*) : JsonObject에서 데이터를 숫자 형식으로 반환하는 API. 2번째 파라미터에 Key를 전달하면 gint64 형식으로 반환합니다.

`JsonNode* json_object_get_member(JsonObject *, gchar*)` : 특정 JsonObject를 반환하는 API. 파라미터는 순서대로 JsonParser, 멤버 이름입니다. 반환 형식은 JsonNode 입니다.

68. XML 파싱

모바일 앱이 서버와 HTTP 통신을 할때 XML 형식의 데이터가 전달됩니다. XML은 데이터를 체계적으로 저장하기 위한 문서 규약으로서 통신 뿐만 아니라 앱개발에 필요한 데이터 저장 (화면 레이아웃, 리소스 데이터)에도 널리 사용되고 있습니다. 이번 예제에서는 XML 데이터를 파싱하는 방법을 알아보겠습니다.

1) XML 데이터 구하기

새로운 소스 프로젝트를 생성하고 Project name을 XmlParse으로 지정합니다. src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "xmlparse.h"
#include <libxml/HTMLparser.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label;
    bool value_begin;
    int value_type;
    char buffer[1024];
} appdata_s;
```

libxml/HTMLparser.h는 XML 파싱을 위한 라이브러리 헤더파일입니다.

value_begin는 데이터 시작 여부를 저장하기 위한 변수입니다.

value_type는 데이터 종류를 저장하기 위한 변수입니다.

buffer[]는 파싱된 데이터를 저장하기 위한 문자 배열 변수입니다.

아래에 간단한 XML 구문이 있습니다. name이 Node 이름이고 Elsa가 데이터에 해당합니다. 이것을 파싱해 보겠습니다.

```
<name>Elsa</name><math>95</math>
```

create_base_gui() 함수 위에 새로운 함수를 생성합니다. Box에 위젯을 추가하는 함수입니다.

```
static void  
my_box_pack(Evas_Object *box, Evas_Object *child, double h_weight, double v_weight,  
           double h_align, double v_align)  
{  
    /* create a frame we shall use as padding around the child widget */  
    Evas_Object *frame = elm_frame_add(box);  
    /* use the medium padding style. there is "pad_small", "pad_medium",  
     * "pad_large" and "pad_huge" available as styles in addition to the  
     * "default" frame style */  
    elm_object_style_set(frame, "pad_medium");  
    /* set the input weight/aling on the frame instead of the child */  
    evas_object_size_hint_weight_set(frame, h_weight, v_weight);  
    evas_object_size_hint_align_set(frame, h_align, v_align);  
    {  
        /* tell the child that is packed into the frame to be able to expand */
```

```

        evas_object_size_hint_weight_set(child, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
        /* fill the expanded area (above) as opposed to center in it */
        evas_object_size_hint_align_set(child, EVAS_HINT_FILL,
EVAS_HINT_FILL);
        /* actually put the child in the frame and show it */
        evas_object_show(child);
        elm_object_content_set(frame, child);
    }
    /* put the frame into the box instead of the child directly */
    elm_box_pack_end(box, frame);
    /* show the frame */
    evas_object_show(frame);
}

```

그런 다음 `create_base_gui()` 함수에 Box와 Button 생성 코드를 추가합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

Evas_Object *box, *btn;

/* Box */
box = elm_box_add(ad->conform);
elm_box_horizontal_set(box, EINA_FALSE);
evas_object_size_hint_weight_set(box, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND,
```

```

EVAS_HINT_EXPAND);

evas_object_size_hint_align_set(box, EVAS_HINT_FILL, EVAS_HINT_FILL);
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{

/* Label */
ad->label = elm_label_add(ad->conform);
elm_object_text_set(ad->label, "<align=center>Hello EFL</>");
my_box_pack(box, ad->label, 1.0, 1.0, -1.0, -1.0);

/* Button-1 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Parse1");
evas_object_smart_callback_add(btn, "clicked", btn_parse1_cb, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 1.0);

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

}

```

create_base_gui() 함수 위에 새로운 함수 2개를 추가합니다.

```

void
walkTree1(xmlNode * a_node, appdata_s *ad)
{
    xmlNode *cur_node = NULL;
    xmlChar *key = NULL;
    xmlChar *value = NULL;

    for (cur_node = a_node; cur_node; cur_node = cur_node->next)
    {

```

```

if(!strcmp((const char*)cur_node->name, "name"))
    ad->value_type = 1;
if(!strcmp((const char*)cur_node->name, "math"))
    ad->value_type = 2;

if(!strcmp((const char*)cur_node->name, "text")) {
    if( ad->value_type == 1 )
    {
        value = cur_node->content;
        strcat(ad->buffer, "Name : ");
        strcat(ad->buffer, (char*)value);
        ad->value_type = 0;
    }
    else if( ad->value_type == 2 )
    {
        value = cur_node->content;
        strcat(ad->buffer, " / Math : ");
        strcat(ad->buffer, (char*)value);
        strcat(ad->buffer, "<br/>");
        ad->value_type = 0;
    }
}

walkTree1(cur_node->children, ad);
}

static void
btn_parse1_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    const char* buf = "<name>Elsa</name><math>95</math>";

    htmlParserCtxtPtr parser = htmlCreatePushParserCtxt(NULL, NULL, NULL, 0, NULL, 0);
    htmlCtxtUseOptions(parser, HTML_PARSE_NOBLANKS | HTML_PARSE_NOERROR |

```

```

HTML_PARSE_NOWARNING | HTML_PARSE_NONET);
htmlParseChunk(parser, buf, strlen(buf), 0);

ad->buffer[0] = '₩';
ad->value_type = 0;
walkTree1(xmlDocGetRootElement(parser->myDoc), ad);
elm_object_text_set(ad->label, ad->buffer);
}

```

`walkTree1()` 는 `xmlNode` 객체에서 "name" 과 "math"에 해당하는 데이터를 추출해서 전역변수에 추가하는 함수입니다.

`for` 루프에서 XML 구문에 포함된 개별 노드를 구해서 노드 이름이 'name' 혹은 'math' 인 경우에는 데이터를 구합니다. 다음 번 노드로 넘어가기 위해서 자신을 재귀호출 합니다.

`xmlNode` 는 XML 구문에서 한 지점을 가리키고 있는 구조체 변수입니다. 속성은 다음과 같습니다.

- `next` : 다음 번 노드의 포인터를 반환합니다.
- `name` : 노드 이름. "text"가 저장되어 있는 경우에는 데이터를 의미합니다. 이런 경우에는 `content` 속성에서 데이터를 구할 수 있습니다.
- `content` : 노드 데이터.
- `properties` : 노드 속성 데이터.

`btn_parse1_cb()` 는 XML 파서 객체를 생성해서 XML 구문을 입력하고 파싱 수행 함수를 호출하는 함수입니다.

`htmlCreatePushParserCtxt(htmlSAXHandlerPtr sax, void *user_data, char *chunk, int size, char *filename, xmlCharEncoding enc)` 는 XML 파서 객체를 생성하는 API입니다.

`htmlParserCtxtPtr` 는 XML 파서 구조체입니다. 속성 중에서 `myDoc` 는 XML 구문이 `xmlDocPtr` 형식으로 저장되어 있습니다.

`htmlCtxtUseOptions(htmlParserCtxtPtr ctxt, int options)` 는 XML 파서에 옵션을 지정하는 API입니다. 옵션은 중복 지정할 수 있습니다.

`htmlParseChunk(htmlParserCtxtPtr ctxt, char *chunk, int size, int terminate)` 는 XML 파서에 XML 구문을 입력하는 API 입니다.

`xmlDocGetRootElement(xmlDocPtr doc)` 는 XML 구문의 시작 노드를 반환하는 API 입니다.

예제를 빌드하고 실행시켜 보겠습니다. Button을 누르면 `name` 노드의 데이터와 `math` 노드의 데이터가 화면에 출력됩니다.



2) 노드 속성 데이터 구하기

XML 은 아래와 같이 노드에 속성을 지정할 수 있습니다. 이번에는 노드 속성 데이터를 구하는 방법을 알아보겠습니다.

```
<student name="Aurora" math="27"></student>
```

create_base_gui() 함수 끝부분에 2번째 Button 생성 코드를 추가합니다.

```
/* Button-1 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Parse1");
evas_object_smart_callback_add(btn, "clicked", btn_parse1_cb, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 1.0);

/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Parse2");
evas_object_smart_callback_add(btn, "clicked", btn_parse2_cb, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 1.0);
```

create_base_gui() 함수 위에 새로운 함수 2개를 생성합니다.

```
void
walkTree2(xmlDoc *doc, xmlNode * a_node, appdata_s *ad)
{
    xmlNode *cur_node = NULL;
    xmlAttr *cur_attr = NULL;
    xmlChar *key = NULL;
```

```

for (cur_node = a_node; cur_node; cur_node = cur_node->next)
{
    for (cur_attr = cur_node->properties; cur_attr; cur_attr = cur_attr->next) {
        key = xmlGetProp(cur_node, cur_attr->name);

        if(!strcmp((const char*)cur_attr->name, "name"))
        {
            strcat(ad->buffer, "Name : ");
            strcat(ad->buffer, key);
        }
        if(!strcmp((const char*)cur_attr->name, "math"))
        {
            strcat(ad->buffer, " / Math : ");
            strcat(ad->buffer, key);
            strcat(ad->buffer, "<br/>");
        }
    }

    if(key!=NULL) { xmlFree(key); key=NULL; }
}

walkTree2(doc, cur_node->children, ad);
}

static void
btn_parse2_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    const char* buf = "<?xml version=\"1.0\" encoding=\"utf-8\"?> <grade>
<name>M.I.T</name> <student name=\"Aurora\" math=\"27\"></student>
<student name=\"Piana\" math=\"88\"></student> <student name=\"Tangled\" math=\"77\"></student> </grade>";

    htmlParserCtxtPtr parser = htmlCreatePushParserCtxt(NULL, NULL, NULL, 0, NULL, 0);
    htmlCtxtUseOptions(parser, HTML_PARSE_NOBLANKS | HTML_PARSE_NOERROR |
HTML_PARSE_NOWARNING | HTML_PARSE_NONET);

```

```
htmlParseChunk(parser, buf, strlen(buf), 0);
ad->buffer[0] = 'W0';
walkTree2(parser->myDoc, xmlDocGetRootElement(parser->myDoc), ad);
elm_object_text_set(ad->label, ad->buffer);
}
```

walkTree2() 는 XML 노드의 속성 데이터를 구해서 화면에 출력하는 함수입니다.

1번째 for 루프에서 XML 구문에 포함된 개별

노드를 구하고, 2번째 for 루프에서 개별 속성을 구해서 노드 이름이 'name' 혹은 'math' 인 경우에는 데이터를 구합니다. 다음 번 노드로 넘어가기 위해서 자신을 재귀호출 합니다.

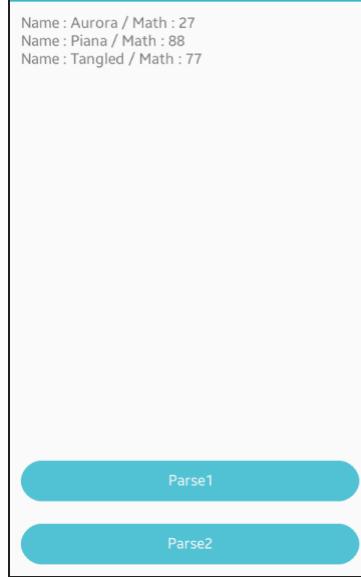
xmlNode의 속성 중에서 properties에는 노드 속성이 xmlAttr 형식으로 저장되어 있습니다.

xmlGetProp(xmlNodePtr node, const xmlChar *name) 는 노드 속성 데이터를 반환하는 API입니다. 1번째 파라미터에는 XML 노드, 2번째 파라미터에는 속성 이름을 전달합니다.

xmlFree(xmlChar*) 는 XML 데이터를 삭제하는 API입니다.

btn_parse2_cb() 는 XML 파서 객체를 생성해서 XML 구문을 입력하고 속성 데이터 파싱 수행 함수를 호출하는 함수입니다.

예제를 다시 실행하고 2번째 Button을 눌러봅시다. 노드 속성 데이터가 화면에 출력됩니다.



3) 다단계 노드 파싱하기

아래 XML 구문에는 총 3개의 <student> 노드가 있고 각각의 <student> 노드 안에는 <name> 와 <math> 노드가 포함되어 있습니다. 트리 구조로 노드에 접근하는 방법을 알아보겠습니다.

```
<student><name>Obama</name><math>50</math></student>
<student><name>Psy</name><math>70</math></student>
<student><name>Yuna</name><math>65</math></student>
```

create_base_gui() 함수 끝부분에 3번째 Button 생성 코드를 추가합니다.

```
/* Button-2 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Parse2");
evas_object_smart_callback_add(btn, "clicked", btn_parse2_cb, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 1.0);
```

```

/* Button-3 */
btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Parse3");
evas_object_smart_callback_add(btn, "clicked", btn_parse3_cb, ad);
my_box_pack(box, btn, 1.0, 0.0, -1.0, 1.0);
}

```

create_base_gui() 함수 위에 새로운 함수 2개를 생성합니다.

```

void
walkTree3(xmlDoc *doc, xmlNode * a_node, appdata_s *ad)
{
    xmlNode *cur_node = NULL;
    xmlAttr *cur_attr = NULL;
    xmlChar *key = NULL;
    xmlChar *value = NULL;

    for (cur_node = a_node; cur_node; cur_node = cur_node->next)
    {
        if(!strcmp((const char*)cur_node->name, "student") )
            ad->value_begin = true;

        if(!strcmp((const char*)cur_node->name, "name") && ad->value_begin)
            ad->value_type = 1;
        if(!strcmp((const char*)cur_node->name, "math") && ad->value_begin)
            ad->value_type = 2;

        if(!strcmp((const char*)cur_node->name, "text")) {
            if( ad->value_type == 1 )
            {
                value = (char*)cur_node->content;
                strcat(ad->buffer, "Name : ");

```

```

        strcat(ad->buffer, (char*)value);
        ad->value_type = 0;
    }
    else if( ad->value_type == 2 )
    {
        value = (char*)cur_node->content;
        strcat(ad->buffer, " / Math : ");
        strcat(ad->buffer, (char*)value);
        strcat(ad->buffer, "<br/>");
        ad->value_type = 0;
    }
}

walkTree3(doc, cur_node->children, ad);
}
}

static void
btn_parse3_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    const char* buf = "<?xml version=\"1.0\" encoding=\"utf-8\"?> <grade>
<name>M.I.T</name>
<student><name>Obama</name><math>50</math></student>
<student><name>Psy</name><math>70</math></student>
<student><name>Yuna</name><math>65</math></student> </grade>";

    htmlParserCtxtPtr parser = htmlCreatePushParserCtxt(NULL, NULL, NULL, 0, NULL, 0);
    htmlCtxtUseOptions(parser, HTML_PARSE_NOBLANKS | HTML_PARSE_NOERROR | HTML_PARSE_NOWARNING | HTML_PARSE_NONET);
    htmlParseChunk(parser, buf, strlen(buf), 0);

    ad->value_begin = false;
    ad->buffer[0] = '\0';
    ad->value_type = 0;
}

```

```
    walkTree3(parser->myDoc, xmlDocGetRootElement(parser->myDoc), ad);

    elm_object_text_set(ad->label, ad->buffer);
}


```

walkTree3() 는 XML 트리 구조에서 2차 노드의 데이터를 구해서 화면에 출력하는 함수입니다.

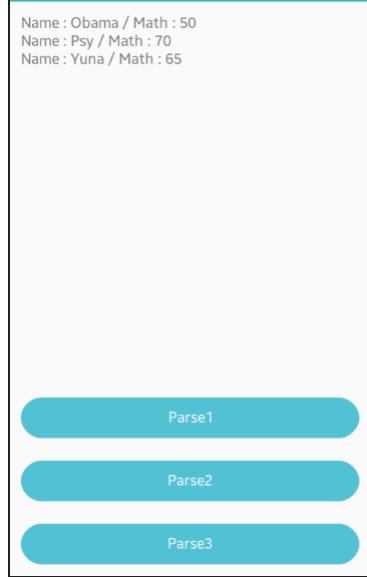
XML 구문에서 노드를 하나씩 추출합니다. 노드 이름이 "student" 이면 1차 노드로 판단하고 전역변수 value_begin을 true로 설정 합니다.

노드 이름이 "name" 또는 "math" 이고 , 전역변수 value_begin이 true라면 2차 노드로 판단하고 자기자신을 재귀호출합니다.

노드 이름이 "text" 이면 노드 데이터 라고 판단하고 content 속성 값을 읽어서 전역변수에 추가합니다. 다음번 노드로 넘어가기 위해서 자신을 재귀호출 합니다.

btn_parse3_cb() 는 XML 파서 객체를 생성해서 XML 구문을 입력하고 2차 노드 데이터 파싱 수행 함수를 호출하는 함수입니다.

예제를 다시 실행하고 3번째 Button을 눌러봅시다. 2차 노드의 데이터가 화면에 출력됩니다.



4) 관련 API

```
htmlParserCtxtPtr htmlCreatePushParserCtxt(htmlSAXHandlerPtr sax, void *user_data, char *chunk, int size, char *filename, xmlCharEncoding enc) : XML 파서 객체를 생성하는 API.
```

htmlParserCtxtPtr : XML 파서 구조체. 속성 중에서 myDoc 는 XML 구문이 xmlDocPtr 형식으로 저장되어 있습니다.

```
int htmlCtxtUseOptions(htmlParserCtxtPtr ctxt, int options) : XML 파서에 옵션을 지정하는 API. 옵션은 중복 지정할 수 있습니다.
```

```
int htmlParseChunk(htmlParserCtxtPtr ctxt, char *chunk, int size, int terminate) : XML 파서에 XML 구문을 입력하는 API.
```

```
xmlNodePtr xmlDocGetRootElement(xmlDocPtr doc) : XML 구문의 시작 노드를 반환하는 API.
```

`xmlChar* xmlGetProp(xmlNodePtr node, const xmlChar *name)` : 노드 속성 데이터를 반환하는 API. 1번째 파라미터에는 XML 노드, 2번째 파라미터에는 속성 이름을 전달합니다.

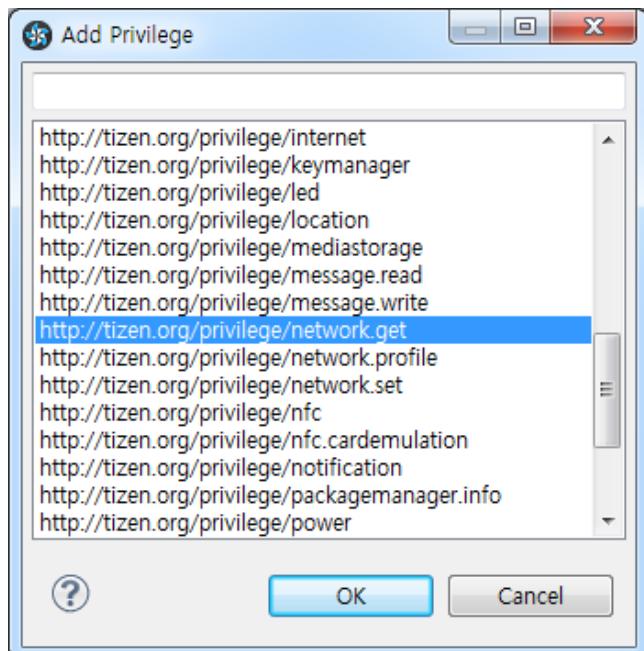
`void xmlFree(xmlChar*)` : XML 데이터를 삭제하는 API.

69. Network 상태 확인

서버와 통신하기 전에 먼저 네트워크 상태를 확인하는 것이 좋습니다. 이번 시간에는 통신 가능 여부를 알아보고 모바일 통신과 WiFi 가능 여부를 확인하는 예제를 만들어 보겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 NetConnection 으로 지정합니다. 통신 상태를 확인하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/privilege/network.get>을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



저장하고 아래쪽 탭버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml

버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<manifest xmlns="http://tizen.org/ns/packages" api-version="2.3"
package="org.example.netconnection" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.netconnection" exec="netconnection"
multiple="false" nodisplay="false" taskmanage="true" type="capp">
        <label>netconnection</label>
        <icon>netconnection.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/network.get</privilege>
    </privileges>
</manifest>
```

2) 통신 연결 상태 확인

통신 연결 상태인지 여부를 확인해 보겠습니다. src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "netconnection.h"
#include <net_connection.h>

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
Evas_Object *label1;
Evas_Object *label2;
Evas_Object *label3;
```

```
connection_h connection;  
} appdata_s;
```

net_connection.h는 통신 상태를 확인하기 위한 라이브러리 헤더파일입니다.

label1에는 통신 연결 상태를 표시하고, label2에는 모바일 통신 상태를 표시하고, label3에는 WiFi 상태를 표시하겠습니다.

connection_h는 통신 정보 구조체 입니다.

create_base_gui() 위 새로운 함수를 생성합니다. 통신 연결 상태를 확인해서 화면에 출력하는 함수입니다.

```
static int  
net_state(appdata_s *ad)  
{  
    int error_code;  
  
    error_code = connection_create(&ad->connection);  
    if (error_code != CONNECTION_ERROR_NONE) {  
        dlog_print(DLOG_ERROR, "tag", "connection error");  
        return error_code;  
    }  
  
    connection_type_e net_state;  
    error_code = connection_get_type(ad->connection, &net_state);  
    switch( net_state )  
    {  
        case CONNECTION_TYPE_DISCONNECTED : /*< Disconnected */  
            elm_object_text_set(ad->label1, "Net state Disconnected");  
            break;  
    }  
}
```

```

        case CONNECTION_TYPE_WIFI : /**< Wi-Fi type */
            elm_object_text_set(ad->label1, "Net state Wifi");
            break;
        case CONNECTION_TYPE_CELLULAR : /**< Cellular type */
            elm_object_text_set(ad->label1, "Net state Cellular");
            break;
        case CONNECTION_TYPE_ETHERNET : /**< Ethernet type */
            elm_object_text_set(ad->label1, "Net state Ethernet");
            break;
        case CONNECTION_TYPE_BT : /**< Bluetooth type */
            elm_object_text_set(ad->label1, "Net state BT");
            break;
    }
    return error_code;
}

```



`connection_create(connection_h* connection)` 는 `connection_h` 객체를 생성하는 API입니다.

`connection_get_type(connection_h connection, connection_type_e* type)` 는 현재 통신 상태를 반환하는 API입니다. 반환하는 형식은 `connection_type_e`입니다. `connection_type_e`의 종류는 다음과 같습니다.

- `CONNECTION_TYPE_DISCONNECTED` : 통신 접속 해제
- `CONNECTION_TYPE_WIFI` : WiFi 형식
- `CONNECTION_TYPE_CELLULAR` : 모바일 통신 형식
- `CONNECTION_TYPE_ETHERNET` : 이더넷 형식
- `CONNECTION_TYPE_BT` : 블루투스 형식

`create_base_gui()` 함수의 코드를 수정합니다. Frame, Box, Label 위젯을 생성하고, 위 함수를 호출하는 코드입니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Frame for outer padding */
Evas_Object *frame = elm_frame_add(ad->win);
elm_object_style_set(frame, "pad_huge");
elm_object_content_set(ad->conform, frame);
evas_object_show(frame);

/* Vertical box */
Evas_Object *vbox = elm_box_add(ad->win);
elm_box_padding_set(vbox, ELM_SCALE_SIZE(10), ELM_SCALE_SIZE(10));
elm_object_content_set(frame, vbox);
evas_object_show(vbox);

{
    /* Label-1 */
    ad->label1 = elm_label_add(ad->conform);
    elm_object_text_set(ad->label1, "Net state");
    evas_object_size_hint_weight_set(ad->label1, EVAS_HINT_EXPAND, 0);
    elm_box_pack_end(vbox, ad->label1);
    evas_object_show(ad->label1);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

```

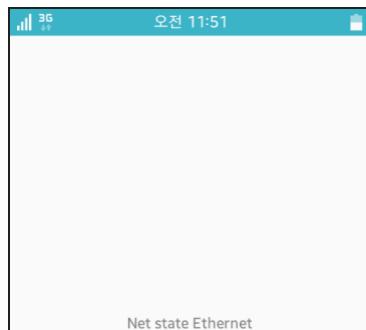
```
    int error_code = net_state(ad);  
}  
_____
```

앱이 종료될 때 connection_h 객체를 삭제해 줍니다. app_terminate() 함수에 새로운 코드를 추가합니다.

```
_____  
static void  
app_terminate(void *data)  
{  
    appdata_s *ad = data;  
    connection_destroy(ad->connection);  
}
```

connection_destroy(connection_h connection) 는 connection_h 객체를 삭제하는 API입니다.

예제를 빌드하고 실행시켜 봅시다. 통신 연결 종류가 Label 위젯에 표시됩니다.



2) 모바일 통신 상태 구하기

유료 통신인 모바일 통신의 접속 상태를 구해 보겠습니다.
create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```
static void
cellular_state(appdata_s *ad)
{
    int error_code;
    connection_cellular_state_e state;
    error_code = connection_get_cellular_state(ad->connection, &state);

    switch( state )
    {
        case CONNECTION_CELLULAR_STATE_OUT_OF_SERVICE:
            elm_object_text_set(ad->label2, "Cell state Out of service");
            break;
        case CONNECTION_CELLULAR_STATE_FLIGHT_MODE:
            elm_object_text_set(ad->label2, "Cell state Flight mode");
            break;
        case CONNECTION_CELLULAR_STATE_ROAMING_OFF:
            elm_object_text_set(ad->label2, "Cell state Roaming off");
            break;
        case CONNECTION_CELLULAR_STATE_CALL_ONLY_AVAILABLE:
            elm_object_text_set(ad->label2, "Cell state Call only");
            break;
        case CONNECTION_CELLULAR_STATE_AVAILABLE:
            elm_object_text_set(ad->label2, "Cell state Available");
            break;
        case CONNECTION_CELLULAR_STATE_CONNECTED:
            elm_object_text_set(ad->label2, "Cell state Connected");
            break;
        default:
    }
}
```

```
    elm_object_text_set(ad->label2, "Cell state Error");
    break;
}
}
```

connection_get_cellular_state(connection_h connection,
connection_cellular_state_e* state) 는 모바일 통신 상태를 반환하는
API입니다. 반환하는 형식은 connection_cellular_state_e입니다.
connection_cellular_state_e의 종류는 다음과 같습니다.

- CONNECTION_CELLULAR_STATE_OUT_OF_SERVICE : 접속 해제
- CONNECTION_CELLULAR_STATE_FLIGHT_MODE : 비행 모드
- CONNECTION_CELLULAR_STATE_ROAMING_OFF : 로밍 Off
- CONNECTION_CELLULAR_STATE_CALL_ONLY_AVAILABLE : 음성 통화만 가능
- CONNECTION_CELLULAR_STATE_AVAILABLE : 사용 가능하지만 아직 접속 되지는 않은 상태
- CONNECTION_CELLULAR_STATE_CONNECTED : 접속 상태

새로운 Label 위젯을 추가해서 위 함수의 결과를 출력해 보겠습니다.
create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Label-1 */
ad->label1 = elm_label_add(ad->conform);
elm_object_text_set(ad->label1, "Net state");
evas_object_size_hint_weight_set(ad->label1, EVAS_HINT_EXPAND, 0);
elm_box_pack_end(vbox, ad->label1);
evas_object_show(ad->label1);

/* Label-2 */
ad->label2 = elm_label_add(ad->conform);
```

```

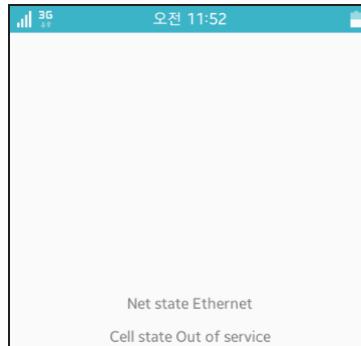
elm_object_text_set(ad->label2, "Cell state");
evas_object_size_hint_weight_set(ad->label2, EVAS_HINT_EXPAND, 0);
elm_box_pack_end(vbox, ad->label2);
evas_object_show(ad->label2);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

int error_code = net_state(ad);
if (error_code == CONNECTION_ERROR_NONE) {
    cellular_state(ad);
}
}

```

예제를 다시 실행시켜 봅시다. 모바일 통신 상태가 2번째 Label 위젯에 출력됩니다.



3) WiFi 상태 구하기

WiFi 접속 상태를 구해 보겠습니다. `create_base_gui()` 함수 위에 새로운 함수를 생성합니다.

```

static void
wifi_state(appdata_s *ad)
{
    connection_wifi_state_e wifi_state;
    connection_get_wifi_state(ad->connection, &wifi_state);
    switch (wifi_state)
    {
        case CONNECTION_WIFI_STATE_DEACTIVATED:
            elm_object_text_set(ad->label3, "Wifi state Deactivated");
            break;
        case CONNECTION_WIFI_STATE_DISCONNECTED:
            elm_object_text_set(ad->label3, "Wifi state Disconnected");
            break;
        case CONNECTION_WIFI_STATE_CONNECTED:
            elm_object_text_set(ad->label3, "Wifi state Connected");
            break;
        default:
            dlog_print(DLOG_INFO, "tag", "Wifi error");
            elm_object_text_set(ad->label3, "Wifi state Error");
            break;
    }
}

```

connection_get_wifi_state(connection_h connection,
 connection_wifi_state_e* state) 는 WiFi 통신 상태를 반환하는 API입니다.
 반환하는 형식은 connection_wifi_state_e입니다.
 connection_wifi_state_e의 종류는 다음과 같습니다.

- CONNECTION_WIFI_STATE_DEACTIVATED : WiFi 비활성화 상태
- CONNECTION_WIFI_STATE_DISCONNECTED : WiFi 연결 해제 상태
- CONNECTION_WIFI_STATE_CONNECTED : WiFi 접속 상태

새로운 Label 위젯을 추가해서 위 함수의 결과를 출력해 보겠습니다.
create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Label-2 */
ad->label2 = elm_label_add(ad->conform);
elm_object_text_set(ad->label2, "Cell state");
evas_object_size_hint_weight_set(ad->label2, EVAS_HINT_EXPAND, 0);
elm_box_pack_end(vbox, ad->label2);
evas_object_show(ad->label2);

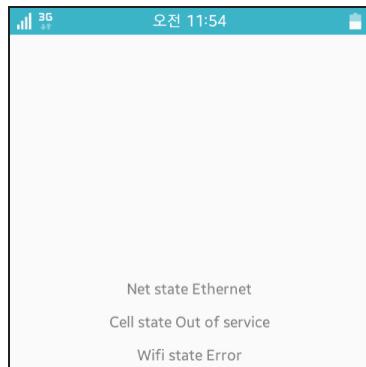
/* Label-3 */
ad->label3 = elm_label_add(ad->conform);
elm_object_text_set(ad->label3, "Wifi state");
evas_object_size_hint_weight_set(ad->label3, EVAS_HINT_EXPAND, 0);
elm_box_pack_end(vbox, ad->label3);
evas_object_show(ad->label3);

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

int error_code = net_state(ad);
if (error_code == CONNECTION_ERROR_NONE) {
    cellular_state(ad);
    wifi_state(ad);
}
}
```

예제를 다시 실행시켜 봅시다. 현재 WiFi 접속 상태가 3번째 Label 위젯에 출력됩니다.



4) 관련 API

`int connection_create(connection_h* connection)` 는 `connection_h` 객체를 생성하는 API입니다.

`int connection_get_type(connection_h connection, connection_type_e* type)` 는 현재 통신 상태를 반환하는 API입니다. 반환하는 형식은 `connection_type_e`입니다. `connection_type_e`의 종류는 다음과 같습니다.

- `CONNECTION_TYPE_DISCONNECTED` : 통신 접속 해제
- `CONNECTION_TYPE_WIFI` : WiFi 형식
- `CONNECTION_TYPE_CELLULAR` : 모바일 통신 형식
- `CONNECTION_TYPE_ETHERNET` : 이더넷 형식
- `CONNECTION_TYPE_BT` : 블루투스 형식

`int connection_destroy(connection_h connection)` 는 `connection_h` 객체를 삭제하는 API입니다.

`int connection_get_cellular_state(connection_h connection, connection_cellular_state_e* state)` 는 모바일 통신 상태를 반환하는 API입니다. 반환하는 형식은 `connection_cellular_state_e`입니다. `connection_cellular_state_e`의 종류는 다음과 같습니다.

- `CONNECTION_CELLULAR_STATE_OUT_OF_SERVICE` : 접속 해제
- `CONNECTION_CELLULAR_STATE_FLIGHT_MODE` : 비행 모드
- `CONNECTION_CELLULAR_STATE_ROAMING_OFF` : 로밍 Off
- `CONNECTION_CELLULAR_STATE_CALL_ONLY_AVAILABLE` : 음성 통화만 가능
- `CONNECTION_CELLULAR_STATE_AVAILABLE` : 사용 가능하지만 아직 접속 되지는 않은 상태
- `CONNECTION_CELLULAR_STATE_CONNECTED` : 접속 상태

`int connection_get_wifi_state(connection_h connection, connection_wifi_state_e* state)` 는 WiFi 통신 상태를 반환하는 API입니다. 반환하는 형식은 `connection_wifi_state_e`입니다. `connection_wifi_state_e`의 종류는 다음과 같습니다.

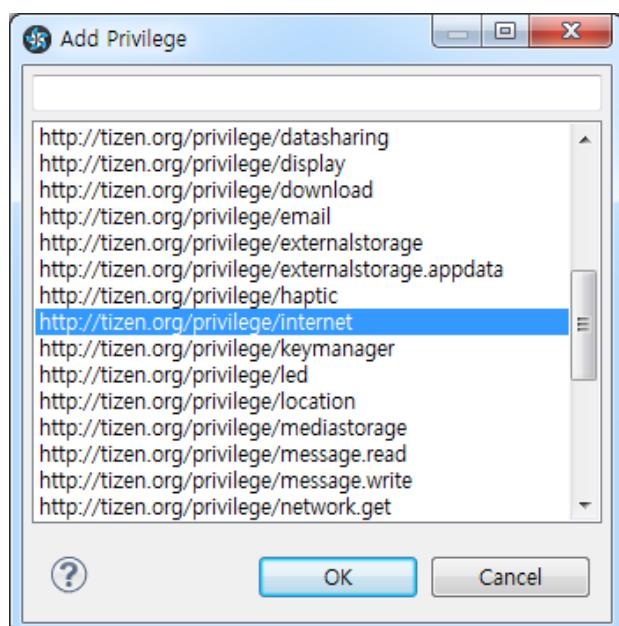
- `CONNECTION_WIFI_STATE_DEACTIVATED` : WiFi 비활성화 상태
- `CONNECTION_WIFI_STATE_DISCONNECTED` : WiFi 연결 해제 상태
- `CONNECTION_WIFI_STATE_CONNECTED` : WiFi 접속 상태

70. Http 통신

HTTP 통신은 사용법이 간편하고 대용량 데이터를 동시에 전송이 가능하기 때문에 모바일에서도 많이 사용되고 있습니다. 이번 예제에서는 HTTP 통신으로 일기예보 정보를 구하고 웹서버에서 이미지를 다운로드 하는 기능을 구현하겠습니다.

1) Privilege 등록

새로운 소스 프로젝트를 생성하고 Project name을 HttpRequest 으로 지정합니다. 통신을 사용하기 위해서는 사용자 권한이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Privileges를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 http://tizen.org/privilege/internet 을 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



저장하고 아래쪽 탭버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest           xmlns="http://tizen.org/ns/packages"           api-version="2.3"
package="org.example.httprequest" version="1.0.0">
    <profile name="mobile"/>
    <ui-application appid="org.example.httprequest" exec="httprequest" multiple="false"
nodisplay="false" taskmanage="true" type="capp">
        <label>httprequest</label>
        <icon>httprequest.png</icon>
    </ui-application>
    <privileges>
        <privilege>http://tizen.org/privilege/internet</privilege>
    </privileges>
</manifest>
```

2) 텍스트 통신 데이터 구하기

웹서버에서 텍스트 데이터를 수신해 보겠습니다. src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수, 그리고 구조체를 추가합니다.

```
#include "httprequest.h"
#include <curl/curl.h>

typedef struct MemoryStruct {
```

```
char *memory;
size_t size;
} memoryStruct;

typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
//Evas_Object *label;
    Evas_Object *entry;
    Evas_Object *icon;
    memoryStruct ms;
} appdata_s;
```

Http 통신에는 CURL 라이브러리가 사용됩니다. curl/curl.h는 CURL 라이브러리 헤더파일입니다.

memoryStruct는 통신 결과 데이터를 저장하는 구조체입니다.

icon은 Evas Image 객체입니다.

서버에 접속해서 텍스트 데이터를 수신하는 기능을 구현하겠습니다.
create_base_gui() 함수 위에 새로운 함수 3개를 생성합니다.

```
static size_t
write_memory_cb(void *contents, size_t size, size_t nmemb, void *userp)
{
    size_t realsize = size * nmemb;
    memoryStruct *mem = (memoryStruct *)userp;

    mem->memory = realloc(mem->memory, mem->size + realsize + 1);
    if(mem->memory == NULL) {
        /* out of memory! */
    }
}
```

```

dlog_print(DLOG_INFO, "tag", "not enough memory (realloc returned NULL)");
return 0;
}

memcpy(&(mem->memory[mem->size]), contents, realsize);
mem->size += realsize;
mem->memory[mem->size] = 0;
return realsize;
}

void
get_http_data(const char* url, memoryStruct *data)
{
    CURL *curl_handle;
    CURLcode res;

    data->memory = malloc(1); /* will be grown as needed by the realloc above */
    data->size = 0; /* no data at this point */

    curl_global_init(CURL_GLOBAL_ALL);

    /* init the curl session */
    curl_handle = curl_easy_init();

    /* specify URL to get */
    curl_easy_setopt(curl_handle, CURLOPT_URL, url);

    /* send all data to this function */
    curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION, write_memory_cb);

    /* we pass our 'chunk' struct to the callback function */
    curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, (void *)data);

    /* some servers don't like requests that are made without a user-agent
     * field, so we provide one */

```

```

curl_easy_setopt(curl_handle, CURLOPT_USERAGENT, "libcurl-agent/1.0");

/* get it!
res = curl_easy_perform(curl_handle);

/* cleanup curl stuff */
curl_easy_cleanup(curl_handle);

/* we're done with libcurl, so clean it up */
curl_global_cleanup();
}

static void
btn_download_text(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char url[100]={0,};

    sprintf(url,
"http://api.openweathermap.org/data/2.5/weather?lat=37.498&lon=127.027&units=metric");
    get_http_data(url, &ad->ms);

    elm_object_text_set(ad->label, ad->ms.memory);
    free( ad->ms.memory);
}

```

write_memory_cb() 는 서버 응답을 수신받는 이벤트 함수입니다.
파라미터는 순서대로 컨텐츠, Byte 단위, 메모리 크기, 사용자
데이터입니다.

2번째 파라미터와 3번째 파라미터를 곱하면 전체 메모리 크기를 계산할
수 있습니다. Byte 단위는 대부분 1입니다.

그 다음 코드는 `realloc()` 함수를 사용해서 `memoryStruct` 구조체의 `memory` 속성에 메모리를 할당하고, `memcpy()` 함수를 사용해서 데이터를 복사합니다.

그런 다음 `memoryStruct` 구조체의 `size` 속성에 메모리 크기를 더하고, 데이터 끝부분에 0을 대입해서 마지막을 표시합니다.

`get_http_data()` 는 서버에 통신을 시도하는 함수입니다.

`curl_global_init(long flags)` 는 CURL 라이브러리를 초기화하는 API입니다. CURL을 사용하는 앱은 처음에 한번 실행해 주어야 합니다.

`curl_easy_init(void)` 는 CURL 객체를 생성하는 API입니다.

`curl_easy_setopt(CURL *curl, CURLOPT option, ...)` 는 CURL 객체에 옵션을 지정하는 API입니다. 옵션 종류는 다음과 같습니다.

- `CURLOPT_URL` : URL 주소를 지정
- `CURLOPT_WRITEFUNCTION` : 통신 결과 수신 콜백 함수 지정
- `CURLOPT_WRITEDATA` : 사용자 데이터 지정
- `CURLOPT_USERAGENT` : 사용자 Agent 지정

`curl_easy_perform(CURL *curl)` 는 서버와 통신을 시작하는 API 입니다.

`curl_easy_cleanup(CURL *curl)` 는 CURL 데이터를 삭제하는 API입니다.

`curl_global_cleanup(void)` 는 CURL 라이브러리 전체 데이터를 삭제하는 API입니다. CURL을 사용했으면 앱을 종료하기 전에 한번은 호출해 주어야 합니다.

`btn_download_text()` 은 사용자가 Button을 클릭했을 때 서버에서 텍스트 데이터를 수신해서 Label 위젯에 결과를 표시하는 함수입니다.

위도 좌표는 37.498, 경도 좌표는 127.027에 해당하는 위치의 날씨 정보를 요청합니다.

일기예보 서버에서 데이터를 수신하는 Button을 생성하겠습니다.
create_base_gui() 함수 끝부분에 새로운 코드를 추가합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Vertical box */
Evas_Object *vbox = elm_box_add(ad->win);
elm_box_padding_set(vbox, ELM_SCALE_SIZE(10), ELM_SCALE_SIZE(10));
elm_object_content_set(ad->conform, vbox);
evas_object_show(vbox);

{
    /* Entry */
    ad->entry = elm_entry_add(ad->conform);
    evas_object_size_hint_weight_set(ad->entry, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(ad->entry, EVAS_HINT_FILL, EVAS_HINT_FILL);
    elm_box_pack_end(vbox, ad->entry);
    evas_object_show(ad->entry);

    /* Button-1 */
    Evas_Object *btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "Text");
```

```

    evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0);
    evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0);
    evas_object_smart_callback_add(btn, "clicked", btn_download_text, ad);
    elm_box_pack_end(vbox, btn);
    evas_object_show(btn);
}

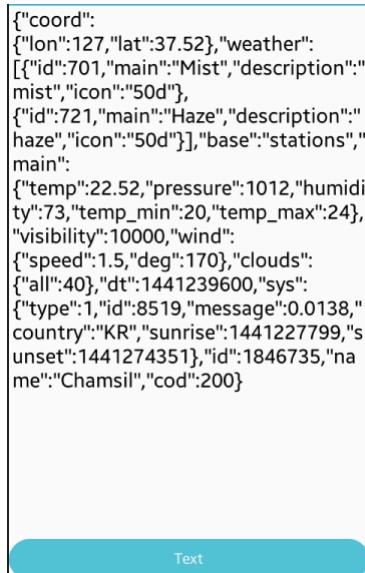
/* Show window after base gui is set up */
evas_object_show(ad->win);

}

```

Box, Entry, Button 위젯을 생성하였습니다.

예제를 빌드하고 실행시켜 보겠습니다. Button을 클릭하고 서버에서 응답이 수신되면 Json 형식의 날씨 정보가 화면에 출력됩니다.



Json 구문에서 개별 데이터를 추출하는 방법은 JsonParse 예제를 참조하기 바랍니다.

3) 이미지 다운로드

이번에는 웹서버에 있는 이미지를 다운로드 해서 Image 객체에 출력하는 기능을 구현해 보겠습니다. `create_base_gui()` 함수에 Image 객체와 Button 위젯 생성 코드를 추가합니다.

```
{  
    /* Entry */  
    ad->entry = elm_entry_add(ad->conform);  
    evas_object_size_hint_weight_set(ad->entry, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
    evas_object_size_hint_align_set(ad->entry, EVAS_HINT_FILL, EVAS_HINT_FILL);  
    elm_box_pack_end(vbox, ad->entry);  
    evas_object_show(ad->entry);  
  
    /* Image */  
    ad->icon = elm_image_add(ad->conform);  
    evas_object_size_hint_weight_set(ad->icon, EVAS_HINT_EXPAND,  
EVAS_HINT_EXPAND);  
    evas_object_size_hint_align_set(ad->icon, EVAS_HINT_FILL, EVAS_HINT_FILL);  
    elm_box_pack_end(vbox, ad->icon);  
    evas_object_show(ad->icon);  
  
    /* Button-1 */  
    Evas_Object *btn = elm_button_add(ad->conform);  
    elm_object_text_set(btn, "Text");  
    evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0);  
    evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0);  
    evas_object_smart_callback_add(btn, "clicked", btn_download_text, ad);  
    elm_box_pack_end(vbox, btn);  
    evas_object_show(btn);  
  
    /* Button-2 */
```

```

btn = elm_button_add(ad->conform);
elm_object_text_set(btn, "Image");
evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0);
evas_object_smart_callback_add(btn, "clicked", btn_download_image, ad);
elm_box_pack_end(vbox, btn);
evas_object_show(btn);
}

```

Image 객체를 생성해서 appdata 구조체의 icon 변수에 저장하였습니다.
그리고 2번째 Button을 생성해서 콜백 함수명을 btn_download_image 으로 지정하였습니다.

마지막으로 Button 콜백 함수를 생성할 차례입니다. create_base_gui() 함수 위에 새로운 함수를 추가합니다.

```

static void
btn_download_image(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char url[100]={0,};

    sprintf(url, "https://www.tizen.org/sites/all/themes/tizen_theme/logo.png");
    get_http_data(url, &ad->ms);

    // update icon image.
    if ( elm_image_memfile_set( ad->icon, ad->ms.memory , ad->ms.size, "png", 0 ) ==
EINA_FALSE)
    {
        dlog_print(DLOG_DEBUG, "tag", "%s : image setting error " , __func__);
    }
    free( ad->ms.memory);
}

```

```
}
```

elm_image_memfile_set(Evas_Object *obj, const void *img, size_t size, const char *format, const char *key) 는 Evas Image 객체에 원본 이미지 데이터를 입력하는 API입니다. 파라미터는 순서대로 Evas Image 객체, 이미지 데이터, 데이터 크기, 이미지 형식입니다.

예제를 빌드하고 2번째 Button을 눌러 봅시다. 웹서버에서 다운 받은 타이젠 로고 이미지가 화면에 표시됩니다.



4) 관련 API

CURLcode curl_global_init(long flags) : CURL 라이브러리를 초기화하는 API. CURL을 사용하는 앱은 처음에 한번 실행해 주어야 합니다.

CURL* curl_easy_init(void) : CURL 객체를 생성하는 API.

CURLcode curl_easy_setopt(CURL *curl, CURLOPT option, ...) : CURL 객체에 옵션을 지정하는 API. 옵션 종류는 다음과 같습니다.

- CURLOPT_URL : URL 주소를 지정
- CURLOPT_WRITEFUNCTION : 통신 결과 수신 콜백 함수 지정
- CURLOPT_WRITEDATA : 사용자 데이터 지정
- CURLOPT_USERAGENT : 사용자 Agent 지정

CURLcode curl_easy_perform(CURL *curl) : 서버와 통신을 시작하는 API.

void curl_easy_cleanup(CURL *curl) : CURL 데이터를 삭제하는 API.

void curl_global_cleanup(void) : CURL 라이브러리 전체 데이터를 삭제하는 API. CURL을 사용했으면 앱을 종료하기 전에 한번은 호출해 주어야 합니다.

Eina_Bool elm_image_memfile_set(Evas_Object *obj, const void *img, size_t size, const char *format, const char *key) : Evas Image 객체에 원본 이미지 데이터를 입력하는 API. 파라미터는 순서대로 Evas Image 객체, 이미지 데이터, 데이터 크기, 이미지 형식입니다.

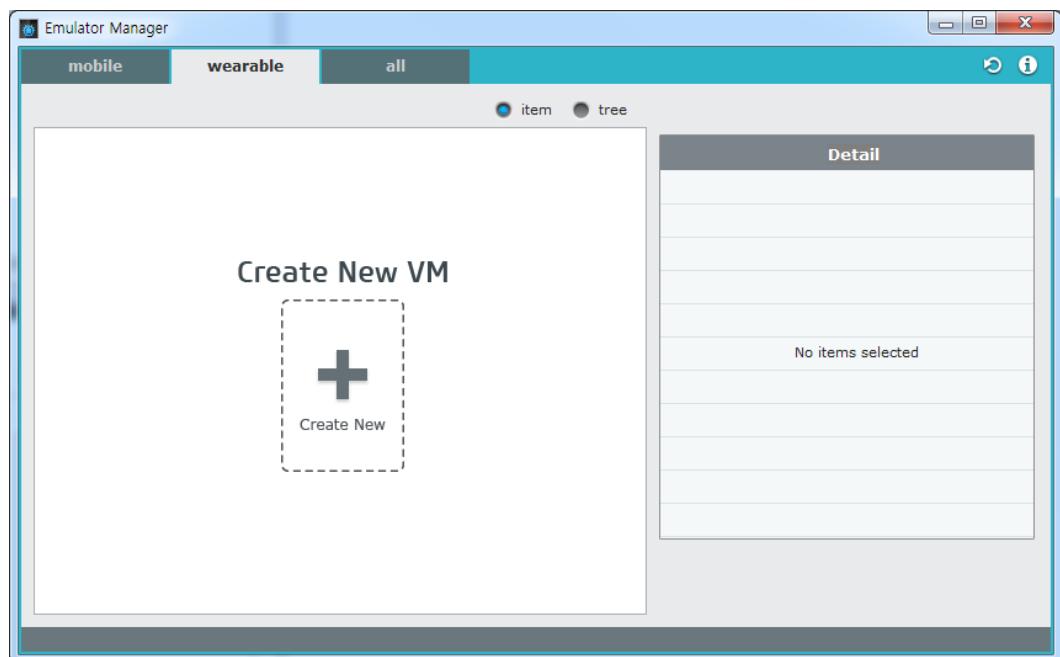
71. Wearable 소스 프로젝트 생성

이번 시간에는 Wearable 에뮬레이터를 생성하고 Wearable 소스 프로젝트를 생성하는 방법을 알아보겠습니다.

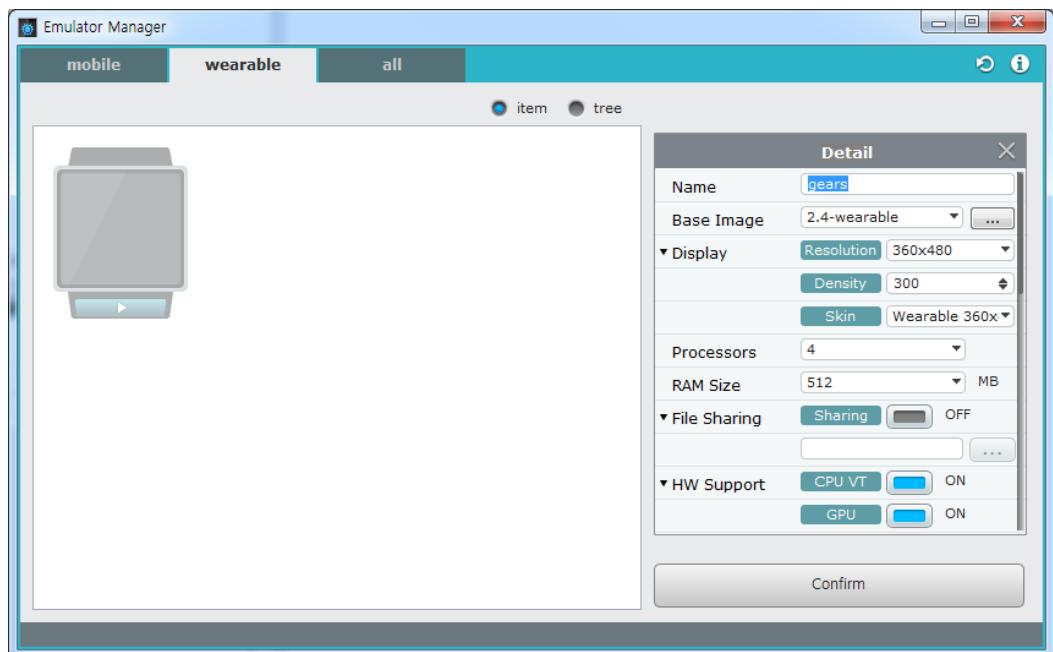
가. Wearable 에뮬레이터 실행하기

에뮬레이터를 생성하고 실행하는 방법을 알아보겠습니다. [윈도우 시작버튼 > 모든 프로그램 > Tizen SDK > Emulator Manager]를 선택합니다. 만약 경고창이 나타나면 무시하고 ‘예’ 버튼을 클릭하면 됩니다.

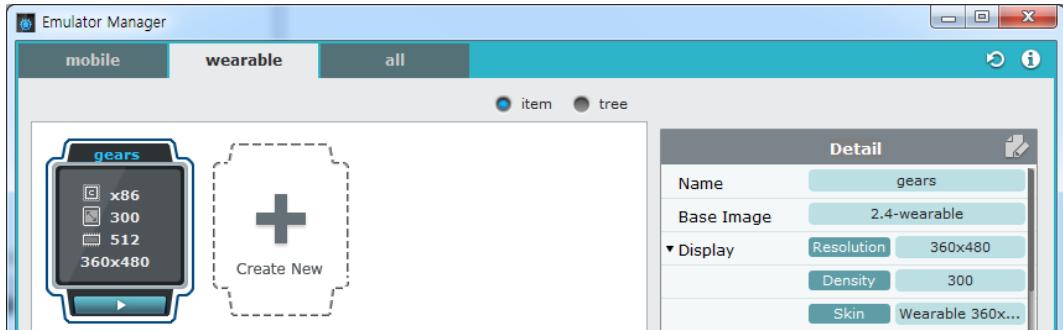
에뮬레이터 매니저가 실행되면 2번째 탭버튼(wearable)을 선택합니다. Wearable 에뮬레이터가 존재하지 않을 때는 에뮬레이터를 생성해야 합니다. 왼쪽 Create New VM 라는 글자 아래에 + 기호를 클릭합니다.



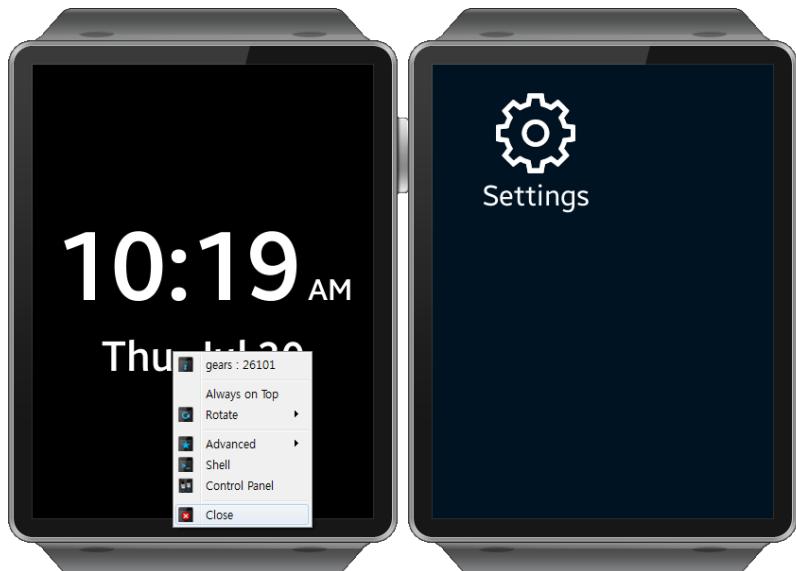
오른쪽에 에뮬레이터의 옵션을 지정하는 화면이 나타납니다. Name 속성에 gears 라고 지정하고 나머지는 디폴트 속성을 그대로 놔두어도 상관없습니다. Confirm 버튼을 클릭하면 에뮬레이터가 생성됩니다.



화면 왼쪽에 gears라는 새로운 에뮬레이터가 생성되었습니다. 새로 생성된 에뮬레이터 아이콘의 아래쪽에 있는 화살표 버튼을 클릭하면 에뮬레이터가 실행됩니다. 만약 에뮬레이터 옵션을 수정하고 싶다면 Reset 버튼을 클릭하면 됩니다.



만약 'Windows 보안 경고' 팝업창이 나타나면 '차단 해제' 버튼을 누르고 계속 진행하면 됩니다. 에뮬레이터 오른쪽 위에는 전원 버튼이 있습니다. Power, Home 2가지 역할을 합니다. 한번 누르면 화면이 On/Off 되고, 오래 누르면 종료됩니다.



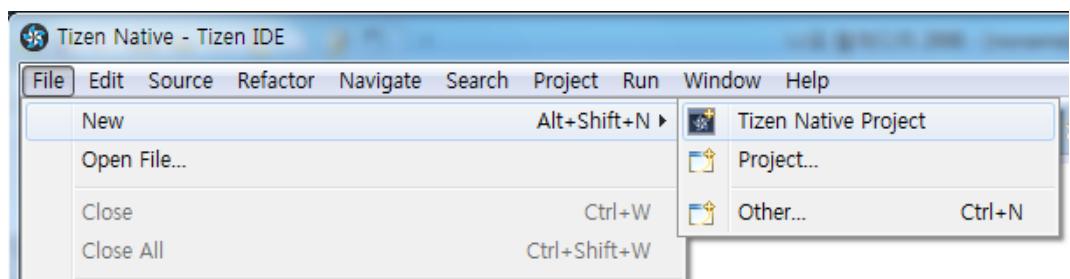
슬립모드로 전환되었을 때는 전원 버튼을 누르거나, 마우스 오른쪽 버튼을 누르고 단축메뉴에서 Close를 선택해도 됩니다.

Flick-Up을 하면 앱 목록이 나타나고, Flick Down을 하면 이전 화면으로 돌아갈수 있습니다.

나. 새로운 소스 프로젝트를 생성해서 Button 클릭 이벤트를 구하는 예제를 만들어 보도록 하겠습니다.

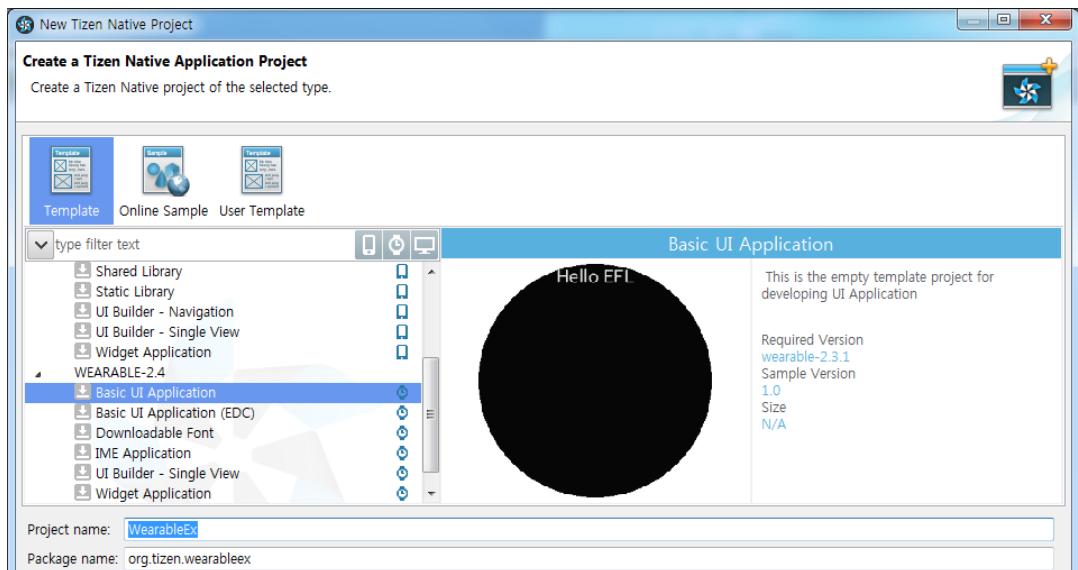
1) 소스 프로젝트 생성

새로운 소스 프로젝트를 생성하겠습니다. 이클립스 메인메뉴 [File > New > Tizen Native Project]를 선택합니다.



소스 프로젝트 생성 팝업창이 나타나면 [Template > WEARABLE-2.x > Basic UI Application]을 선택합니다.

Project name 항목에는 WearableEx 라고 입력합니다.



Package name은 자동으로 입력됩니다. 이제 Finish 버튼을 누르면 새로운 소스 프로젝트가 생성됩니다.

2) 소스 프로젝트 구성요소

Wearable 앱의 기본 소스 프로젝트는 모바일 앱과 거의 유사합니다.

Includes 폴더에는 라이브러리가 포함되어 있습니다.

inc 폴더에는 C언어 헤더파일(~.h) 이 저장되어 있습니다. 라이브러리를 선언하거나 함수 헤더 혹은 전역 변수를 선언할 때 주로 사용합니다.

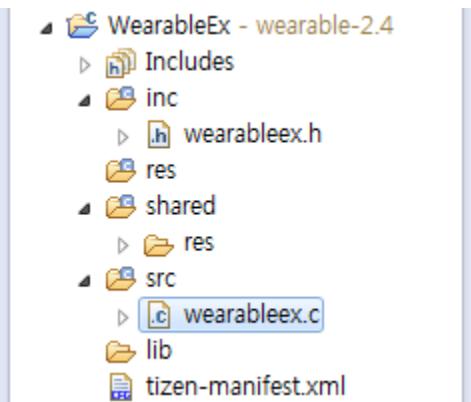
res 폴더에는 이미지나 오디오 파일 같은 리소스 파일을 주로 저장합니다.

src 폴더에는 C언어 소스파일(~.c) 이 저장되어 있습니다. 함수의 기능을 정의할 때 주로 사용하며 대부분의 작업이 여기에서 이루어 집니다.

shared 폴더에는 앱 아이콘 이미지가 저장되어 있습니다. 앱스토어에 앱을 배포할 때 여기에 앱 아이콘을 저장하면 됩니다. 타이젠 스토어는 동그란 모양의 앱아이콘을 사용해야 합니다.

tizen-manifest.xml 파일은 앱의 각종 정보(앱 이름, 버전)와 사용자

권한(Privilege)을 저장하고 있습니다. 안드로이드의 AndroidManifest.xml과 동일한 기능입니다.



3) 기본 소스 프로젝트 실행

처음 소스 프로젝트가 생성된 상태로 실행시켜 보겠습니다.

WearableEx 프로젝트를 마우스 오른쪽 버튼으로 클릭하고, 단축메뉴에서 [Build Project]를 선택합니다. 빌드가 끝나면 다시

WearableEx 프로젝트를 마우스 오른쪽 버튼으로 클릭하고, 단축메뉴에서 [Run As > 1 Tizen Native Application]을 선택합니다. 인증서가 설치되지 않았다면 모바일 에뮬레이터와 동일한 방식으로 인증서를 설치하면 됩니다.

에뮬레이터에 예제가 실행되면 위쪽에 'Hello EFL'이라는 글자가 보입니다. Label 위젯을 사용해서 텍스트를 표시한 것입니다.



4) Label 텍스트 변경

Label 위젯에 표시된 Hello EFL 이라는 텍스트를 변경해 보겠습니다. 그러기 위해서 소스파일을 편집해야 합니다. src 폴더를 열고 wearableex.c 파일을 더블클릭합니다. 이클립스 메인화면에 파일 내용이 보입니다. 소스코드에 대한 설명은 모바일 용 앱과 거의 유사하기 때문에 교재 앞부분에 있는 설명을 참조하기 바랍니다.

화면에 표시된 Hello EFL 이라는 텍스트를 변경해 보겠습니다. `create_base_gui()` 함수의 내용을 아래와 같이 변경합니다.

```
ad->label = elm_label_add(ad->conform);
//elm_object_text_set(ad->label, "<align=center>Hello EFL</align>");
elm_object_text_set(ad->label, "Hello World");
evas_object_size_hint_weight_set(ad->label, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
```

```
elm_object_content_set(ad->conform, ad->label);

/* Show window after base gui is set up */
evas_object_show(ad->win);

}
```

elm_object_text_set() 는 위젯의 캡션 텍스트를 변경하는 API입니다. Label 뿐만 아니라 Button, Entry에도 사용할 수 있습니다. 텍스트 내용에 Html 태그를 사용해서 텍스트 속성을 지정할 수도 있습니다.

다시 실행시켜 보겠습니다. 2번째로 실행할 때는 메인메뉴 [Run > Run]을 누르거나, 단축키 Ctrl + F11을 누르면 됩니다. 예제가 다시 실행되고 화면에 텍스트가 Hello World로 변경되었습니다.



5) Button 위젯 추가

Button을 클릭하면 Label의 텍스트를 변경하는 기능을 구현해 보겠습니다. 방법은 모바일 앱과 동일합니다. create_base_gui() 함수에 새로운 코드를 추가합니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
```

```

evas_object_show(ad->conform);

/* Box */
Evas_Object *box = elm_box_add(ad->win);
elm_box_padding_set(box, ELM_SCALE_SIZE(10), ELM_SCALE_SIZE(10));
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* Label*/
    ad->label = elm_label_add(ad->conform);
    elm_object_text_set(ad->label, "Hello World");
    evas_object_size_hint_weight_set(ad->label,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(ad->label, EVAS_HINT_FILL, 0.0);
    elm_box_pack_end(box, ad->label);
    evas_object_show(ad->label);

    /* Button */
    Evas_Object *btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "Press");
    evas_object_smart_callback_add(btn, "clicked", btn_click_cb, ad);
    evas_object_size_hint_weight_set(btn,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
    evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0.0);
    elm_box_pack_end(box, btn);
    evas_object_show(btn);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);
}

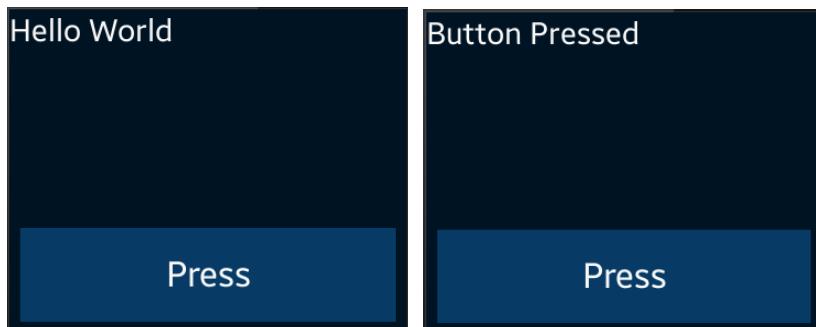
```

Box 컨테이너와 Button 위젯을 생성하는 코드입니다. 클릭 이벤트 콜백

함수명을 `btn_click_cb` 으로 지정했습니다. 이 함수를 만들 차례입니다.
`create_base_gui()` 함수 위에 새로운 함수를 추가합니다.

```
static void  
btn_click_cb(void *data, Evas_Object *obj, void *event_info)  
{  
    appdata_s *ad = data;  
    elm_object_text_set(ad->label, "Button Pressed");  
}  
_____
```

Button을 클릭하면 Label 위젯의 텍스트를 'Button Pressed' 으로
변경하는 내용입니다. 예제를 다시 실행하고 Button을 클릭합시다.
Label의 텍스트가 변경됩니다. Basic UI의 사용방법은 모바일 앱과 거의
유사한 것을 알 수 있습니다.



72. 웨어러블 시스템 정보

교재 앞부분에서 모바일 시스템 정보를 구하는 방법을 알아보았습니다. 이번 시간에는 동일한 앱을 웨어러블 앱으로 제작해서 어떤 부분이 동일하고 어떤 부분이 다른지를 비교해 보겠습니다.

1) 후면 카메라 존재 여부

새로운 소스 프로젝트를 생성하고 형식은 웨어러블 Basic UI Application 방식으로, Project name 은 wSystemInfo 으로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 소스파일 위쪽에 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "systeminfo.h"  
#include <system_info.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label1;  
    Evas_Object *label2;  
    Evas_Object *label3;  
    Evas_Object *label4;  
    Evas_Object *label5;  
} appdata_s;
```

총 5개의 Label 위젯 변수를 선언하였습니다. 1번째 Label에는 후면 카메라 존재 여부, 2번째에는 전화통화 가능 여부, 3번째에는 모니터

수평 픽셀 개수, 4번째에는 모니터 수직 픽셀 개수, 5번째에는 플랫폼 버전을 표시해 보겠습니다.

create_base_gui() 함수 안에 새로운 코드를 추가합니다. 1개의 Button 위젯과 5개의 Label 위젯을 생성하는 코드입니다.

```
/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Box */
Evas_Object *box = elm_box_add(ad->win);
elm_box_padding_set(box, ELM_SCALE_SIZE(10), ELM_SCALE_SIZE(10));
elm_object_content_set(ad->conform, box);
evas_object_show(box);

{
    /* Button */
    Evas_Object *btn = elm_button_add(ad->conform);
    elm_object_text_set(btn, "System Info");
    evas_object_smart_callback_add(btn, "clicked", btn_clicked_cb, ad);
    evas_object_size_hint_weight_set(btn, EVAS_HINT_EXPAND, 0);
    evas_object_size_hint_align_set(btn, EVAS_HINT_FILL, 0);
    elm_box_pack_end(box, btn);
    evas_object_show(btn);
```

```

/* Label-1 */
ad->label1 = elm_label_add(ad->conform);
elm_object_text_set(ad->label1, "Back Camera :");
evas_object_size_hint_weight_set(ad->label1, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(ad->label1, EVAS_HINT_FILL, 0);
elm_box_pack_end(box, ad->label1);
evas_object_show(ad->label1);

/* Label-2 */
ad->label2 = elm_label_add(ad->conform);
elm_object_text_set(ad->label2, "Telephony :");
evas_object_size_hint_weight_set(ad->label2, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(ad->label2, EVAS_HINT_FILL, 0);
elm_box_pack_end(box, ad->label2);
evas_object_show(ad->label2);

/* Label-3 */
ad->label3 = elm_label_add(ad->conform);
elm_object_text_set(ad->label3, "Pixel Width :");
evas_object_size_hint_weight_set(ad->label3, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(ad->label3, EVAS_HINT_FILL, 0);
elm_box_pack_end(box, ad->label3);
evas_object_show(ad->label3);

/* Label-4 */
ad->label4 = elm_label_add(ad->conform);
elm_object_text_set(ad->label4, "Pixel Height :");
evas_object_size_hint_weight_set(ad->label4, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(ad->label4, EVAS_HINT_FILL, 0);
elm_box_pack_end(box, ad->label4);
evas_object_show(ad->label4);

/* Label-5 */
ad->label5 = elm_label_add(ad->conform);
elm_object_text_set(ad->label5, "Platform Ver :");

```

```

    evas_object_size_hint_weight_set(ad->label5, EVAS_HINT_EXPAND, 0);
    evas_object_size_hint_align_set(ad->label5, EVAS_HINT_FILL, 0);
    elm_box_pack_end(box, ad->label5);
    evas_object_show(ad->label5);
}

/* Show window after base gui is set up */
evas_object_show(ad->win);

}

```

Button 콜백 함수를 생성하겠습니다. `create_base_gui()` 함수 위에 새로운 코드를 추가합니다.

```

static void
btn_clicked_cb(void *data, Evas_Object *obj, void *event_info)
{
    appdata_s *ad = data;
    char buf[100];
    char *sValue = NULL;
    bool bValue = false;
    int nValue = 0;
    int ret;

    ret = system_info_get_platform_bool("http://tizen.org/feature/camera.back", &bValue);
    if (ret == SYSTEM_INFO_ERROR_NONE)
    {
        sprintf(buf, "Back Camera : %s", bValue ? "True" : "False");
        elm_object_text_set(ad->label1, buf);
    }
}

```

`system_info_get_platform_bool(char *, bool *)` 는 시스템 정보를 구하는 API 입니다. 반환되는 데이터 형식은 boolean입니다. 1번째 파라미터는 키값이고, "http://tizen.org/feature/camera.back"를 전달하면 후면 카메라 존재 여부를 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 1번째 Label에 텍스트가 변경됩니다. 에뮬레이터에서 실행하면 False가 표시되고, 단말에서 실행하면 True가 표시됩니다.



2) 전화 기능 존재 여부

이번에는 전화 기능이 존재하는지 여부를 확인해 보겠습니다. `btn_clicked_cb()` 함수 끝부분에 새로운 코드를 추가합니다.

```
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "Back Camera : %s", bValue ? "True" : "False");
    elm_object_text_set(ad->label1, buf);
}
```

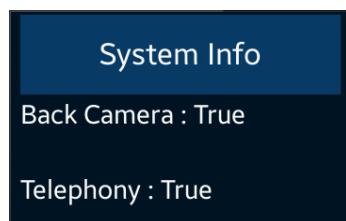
```

ret = 
system_info_get_platform_bool("http://tizen.org/feature/network.telephony",
&nValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "Telephony : %s", bValue ? "True" : "False");
    elm_object_text_set(ad->label2, buf);
}

```

system_info_get_platform_bool() 함수의 1번째 파라미터에 "http://tizen.org/feature/network.telephony"를 전달하면 전화 기능 존재 여부를 반환합니다. 여기서 true 값이 반환된다고 해서 전화 통화나 네트워크를 사용할 수 있다는 의미는 아닙니다. 이것은 어디까지나 하드웨어 통신 기능이 장착되어 있다는 의미입니다. USIM 칩이 없거나 환경설정에서 네트워크 기능을 비활성화 해놓은 상태라면 통신을 사용할 수 없습니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 2번째 Label에 True가 표시됩니다.



4) 모니터 픽셀 개수

이번에는 모니터의 픽셀 개수를 구해 보겠습니다. btn_clicked_cb() 함수

끝부분에 새로운 코드를 추가합니다.

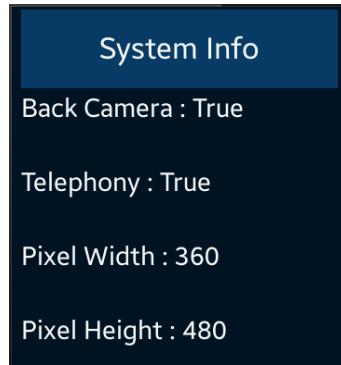
```
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "Telephony : %s", bValue ? "True" : "False");
    elm_object_text_set(ad->label2, buf);
}

ret = system_info_get_platform_int("tizen.org/feature/screen.width", &nValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "Pixel Width : %d", nValue);
    elm_object_text_set(ad->label3, buf);
}

ret = system_info_get_platform_int("tizen.org/feature/screen.height", &nValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "Pixel Height : %d", nValue);
    elm_object_text_set(ad->label4, buf);
}
```

`system_info_get_platform_int(char *, int *)` 는 시스템 정보를 구하는 API입니다. 반환되는 데이터 형식은 정수형 입니다. 1번째 파라미터는 키값이고, "http://tizen.org/feature/screen.width"를 전달하면 모니터 수평 픽셀 개수를 반환합니다. "tizen.org/feature/screen.height"를 전달하면 모니터 수직 픽셀 개수를 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 3번째 Label과 4번째 Label에 숫자가 표시됩니다.



5) 플랫폼 버전

이번에는 플랫폼 버전 정보를 구해 보겠습니다. `btn_clicked_cb()` 함수 끝부분에 새로운 코드를 추가합니다.

```
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "Pixel Height : %d", nValue);
    elm_object_text_set(ad->label4, buf);
}

ret = system_info_get_platform_string("http://tizen.org/feature/platform.version",
&sValue);
if (ret == SYSTEM_INFO_ERROR_NONE)
{
    sprintf(buf, "Platform Ver : %s", sValue);
    elm_object_text_set(ad->label5, buf);
}
```

`system_info_get_platform_string(char *, char **)` 는 시스템 정보를 구하는

API입니다. 반환되는 데이터 형식은 문자열 입니다. 1번째 파라미터는 키값이고, "http://tizen.org/feature/platform.version"를 전달하면 플랫폼 버전을 반환합니다.

예제를 빌드하고 실행시켜 봅시다. Button을 누르면 5번째 Label에 플랫폼 버전이 표시됩니다.



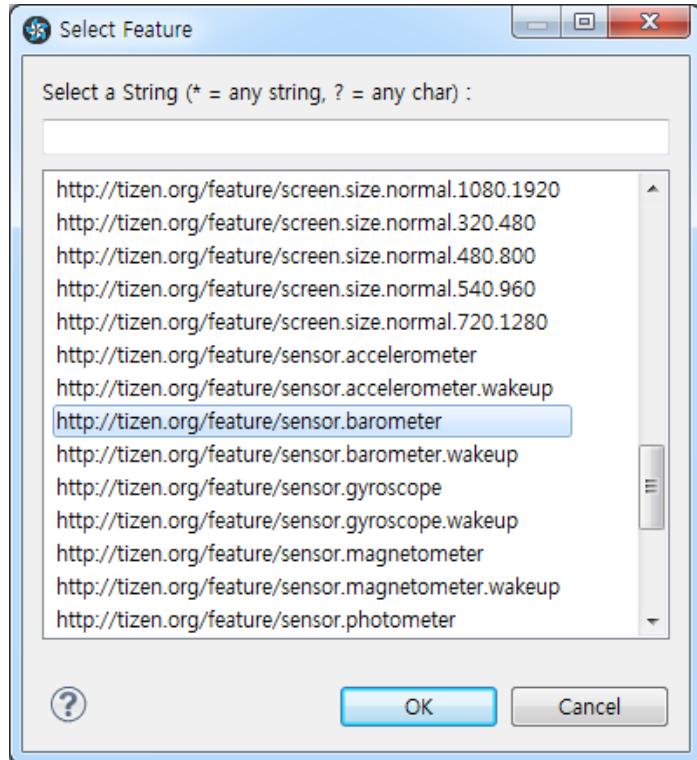
모바일 앱에서 사용했던 API를 웹어러블에서도 그대로 사용할 수 있다는 것을 알수 있습니다.

73. Pressure 센서 사용방법

웨어러블 장비에는 스마트폰에는 없는 심박 센서, 압력 센서 등이 추가됩니다. 이번 예제에서는 압력 센서의 사용방법에 대해서 알아보겠습니다.

1) Feature 등록

새로운 소스 프로젝트를 생성하고 Project name을 wSensorPressure 으로 지정합니다. 압력 센서 기능을 사용하기 위해서는 Feature 등록이 필요합니다. 소스 프로젝트가 생성되었으면 tizen-manifest.xml 파일을 열고 아래쪽 탭버튼 중에서 Features를 누릅니다. 그런 다음 우측 상단에 Add 버튼을 누릅니다. 팝업창이 나타나면 목록에서 <http://tizen.org/feature/sensor.barometer> 를 선택하고 OK 버튼을 눌러서 팝업창을 닫습니다.



동일한 과정을 반복해서 다음의 Feature 를 추가합니다.

- http://tizen.org/feature/sensor.barometer.wakeup

저장하고 아래쪽 텁버튼 중에서 오른쪽 끝에 있는 tizen-manifest.xml 버튼을 누르면 xml 파일의 소스코드가 보입니다.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<manifest      xmlns="http://tizen.org/ns/packages"      api-version="2.3.1"
package="org.example.wsensorpressure" version="1.0.0">
    <profile name="wearable"/>
    <ui-application    appid="org.example.wsensorpressure"    exec="wsensorpressure"
multiple="false" nodisplay="false" taskmanage="true" type="capp">
        <label>wsensorpressure</label>
        <icon>wsensorpressure.png</icon>
    </ui-application>
    <feature name="http://tizen.org/feature/sensor.barometer">true</feature>
```

```
<feature  
name="http://tizen.org/feature/sensor.barometer.wakeup">true</feature>  
</manifest>
```

2) Pressure 센서 지원 여부 판단

새로운 소스 프로젝트를 생성하고 Project name을 wSensorPressure 로 지정합니다. 소스 프로젝트가 생성되었으면 src 폴더에 소스파일(~.c)을 열고 라이브러리 헤더파일과 변수를 추가합니다.

```
#include "wsensorpressure.h"  
#include <sensor.h>  
  
typedef struct appdata {  
    Evas_Object *win;  
    Evas_Object *conform;  
    Evas_Object *label0;  
    Evas_Object *label1;  
} appdata_s;
```

sensor.h는 각종 센서 라이브러리 헤더파일 입니다.

label0에는 Pressure 센서 지원 여부를 표시하고, label1에는 압력 수치를 표시하겠습니다.

create_base_gui() 함수 위에 새로운 함수를 생성합니다.

```
static void show_is_supported(appdata_s *ad)
```

```

{
    char buf[PATH_MAX];
    bool is_supported = false;
    sensor_is_supported(SENSOR_PRESSURE, &is_supported);
    sprintf(buf, "Pressure Sensor is %s", is_supported ? "support" : "not support");
    elm_object_text_set(ad->label0, buf);
}

```

`show_is_supported()` 는 Pressure 센서가 지원되는지를 판단해서 결과를 1번째 Label 위젯에 표시하는 함수입니다.

`sensor_is_supported(sensor_type_e, bool *)` 는 특정 센서의 지원여부를 구하는 API입니다. 1번째 파라미터에 SENSOR_PRESSURE를 전달하면, 2번째 파라미터에서 센서 지원 여부를 반환해 줍니다.

위 함수를 앱이 실행될 때 호출해 주면 됩니다. `create_base_gui()` 함수 끝부분에서 위 함수를 호출합니다.

```

/* Conformant */
ad->conform = elm_conformant_add(ad->win);
elm_win_indicator_mode_set(ad->win, ELM_WIN_INDICATOR_SHOW);
elm_win_indicator_opacity_set(ad->win, ELM_WIN_INDICATOR_OPAQUE);
evas_object_size_hint_weight_set(ad->conform, EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
elm_win_resize_object_add(ad->win, ad->conform);
evas_object_show(ad->conform);

/* Box */
Evas_Object *box = elm_box_add(ad->win);
elm_box_padding_set(box, ELM_SCALE_SIZE(10), ELM_SCALE_SIZE(10));
elm_object_content_set(ad->conform, box);
evas_object_show(box);

```

```

{

/* Label-0 */
ad->label0 = elm_label_add(ad->conform);
elm_label_line_wrap_set(ad->label0, EINA_TRUE);
elm_object_text_set(ad->label0, "Msg - ");
//evas_object_size_hint_weight_set(ad->label,           EVAS_HINT_EXPAND,
EVAS_HINT_EXPAND);
//elm_object_content_set(ad->conform, ad->label);
evas_object_size_hint_weight_set(ad->label0, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(ad->label0, EVAS_HINT_FILL, 0);
elm_box_pack_end(box, ad->label0);
evas_object_show(ad->label0);

/* Label-1 */
ad->label1 = elm_label_add(ad->conform);
elm_label_line_wrap_set(ad->label1, EINA_TRUE);
elm_object_text_set(ad->label1, "Value - ");
evas_object_size_hint_weight_set(ad->label1, EVAS_HINT_EXPAND, 0);
evas_object_size_hint_align_set(ad->label1, EVAS_HINT_FILL, 0);
elm_box_pack_end(box, ad->label1);
evas_object_show(ad->label1);

}

/* Show window after base gui is set up */
evas_object_show(ad->win);

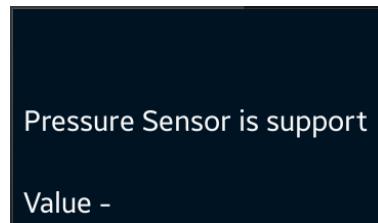
show_is_supported(ad);

}

```

Box 컨테이너와 2개의 Label 위젯을 생성하였습니다. 그리고 센서 지원여부 판단 함수를 호출했습니다.

예제를 빌드하고 실행시켜 봅시다. Pressure 센서가 지원되면 'Pressure Sensor is support'라는 문구가 표시됩니다. 기종에 따라서 센서가 지원되지 않는 경우도 있습니다. 그런 경우에는 에뮬레이터에서 테스트하기 바랍니다.



2) Pressure 센서 이벤트 구하기

Pressure 센서에 물체가 감지되면 그 이벤트를 구해서 거리 수치를 화면에 표시하는 기능을 구현해 보겠습니다. 소스파일 위쪽에 센서 관련 구조체와 전역변수를 추가합니다.

```
typedef struct appdata {
    Evas_Object *win;
    Evas_Object *conform;
    Evas_Object *label0;
    Evas_Object *label1;
} appdata_s;

typedef struct _sensor_info
{
    sensor_h sensor;      /**< Sensor handle */
    sensor_listener_h sensor_listener;
} sensorinfo;

static sensorinfo sensor_info;
```

sensorinfo 는 센서 객체와 이벤트 리스너 변수를 포함하고 있는 구조체입니다.

sensor_info 는 sensorinfo 구조체의 전역변수 입니다.

센서 이벤트를 구하는 것은 리스너를 시작하는 것입니다. 센서 객체와 이벤트 리스너를 사용해서 Pressure 센서 이벤트를 구해보겠습니다. create_base_gui() 함수 위쪽에 새로운 함수 2개를 생성합니다.

```
static void _new_sensor_value(sensor_h sensor, sensor_event_s *sensor_data, void *user_data)
{
    if( sensor_data->value_count < 1 )
        return;
    char buf[PATH_MAX];
    appdata_s *ad = (appdata_s*)user_data;

    sprintf(buf, "Pressure : %0.1f hPa", sensor_data->values[0]);
    elm_object_text_set(ad->label1, buf);
}

static void
start_pressure_sensor(appdata_s *ad)
{
    sensor_error_e err = SENSOR_ERROR_NONE;
    sensor_get_default_sensor(SENSOR_PRESSURE, &sensor_info.sensor);
    err = sensor_create_listener(sensor_info.sensor, &sensor_info.sensor_listener);
    sensor_listener_set_event_cb(sensor_info.sensor_listener, 100, _new_sensor_value, ad);
    sensor_listener_start(sensor_info.sensor_listener);
}
```

_new_sensor_value() 는 Pressure 센서 이벤트 콜백 함수입니다. 새로운

센서 값을 화면에 출력합니다.

2번째 파라미터에 센서 데이터가 전달됩니다. `values[0]`에 수치 데이터가 저장되어 있습니다.

`start_pressure_sensor()` 는 Pressure 센서를 시작하고, 이벤트 콜백 함수를 지정하는 함수입니다.

`sensor_get_default_sensor(sensor_type_e, sensor_h *)` 는 센서 객체를 반환하는 API입니다. 1번째 파라미터에 `SENSOR_PRESSURE` 를 전달하면, 2번째 파라미터에 Pressure 센서 객체가 반환됩니다.

`sensor_create_listener(sensor_h, sensor_listener_h *)` 는 이벤트 리스너를 생성하는 API입니다. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

`sensor_listener_set_event_cb(sensor_listener_h, unsigned int, sensor_event_cb, void *)` 는 리스너에 콜백 함수를 지정하는 API입니다. 파라미터는 순서대로 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

`sensor_listener_start(sensor_listener_h)` 는 리스너를 시작하는 API 입니다.

앱이 실행되면 자동으로 이벤트 리스너를 가동시켜 보겠습니다.

`create_base_gui()` 함수 끝부분에 위 함수를 호출해 줍니다.

```
/* Show window after base gui is set up */  
evas_object_show(ad->win);
```

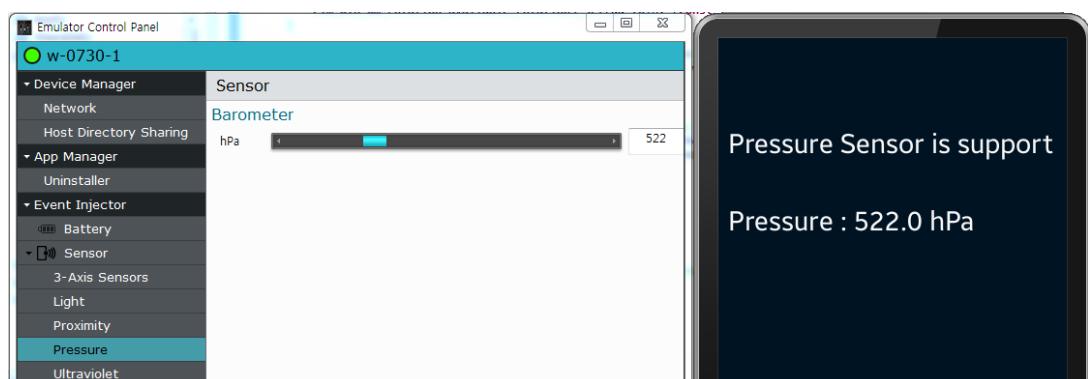
```
show_is_supported(ad);  
start_pressure_sensor(ad);  
}  
_____
```

예제를 다시 실행시켜 보겠습니다. 에뮬레이터에서 테스트 할 때는 Control Panel을 사용하면 됩니다.

에뮬레이터를 마우스 오른쪽 버튼으로 클릭하고 단축 메뉴에서 Control Panel을 선택합니다.

Control Panel이 나타나면 왼쪽 트리 목록에서 [Event Injector > Pressure]를 선택합니다.

Control Panel 오른쪽 화면에서 슬라이드바를 좌우로 드래그 해봅시다. 에뮬레이터 2번째 Label에 수치가 변경되면 정상적으로 구현된 것입니다.



3) 관련 API

int sensor_is_supported(sensor_type_e type, bool *supported) : 특정 센서의 지원여부를 구하는 API. 1번째 파라미터에 SENSOR_PRESSURE를 전달하면, 2번째 파라미터에서 Pressure 센서 지원 여부를 반환해 줍니다.

int sensor_get_default_sensor(sensor_type_e type, sensor_h *sensor) : 센서 객체를 반환하는 API. 1번째 파라미터에 SENSOR_PRESSURE를 전달하면, 2번째 파라미터에 Pressure 센서 객체가 반환됩니다.

int sensor_create_listener(sensor_h sensor, sensor_listener_h *listener) : 이벤트 리스너를 생성하는 API. 1번째 파라미터에 센서 객체를 전달하면, 2번째 파라미터에 리스너 객체가 반환됩니다.

int sensor_listener_set_event_cb(sensor_listener_h listener, unsigned int interval_ms, sensor_event_cb callback, void *data) : 리스너에 콜백 함수를 지정하는 API. / 파라미터 : 이벤트 리스너, 시간 간격(단위는 밀리세컨), 콜백 함수명, 사용자 데이터.

int sensor_listener_start(sensor_listener_h listener) : 리스너를 시작하는 API.