#### Special Interest Group on Innovative Software 2013-2014 Workshop on Android app development

**Session 1: Android App Development - Application Fundamentals** 

#### Contents

1	Setup	development environment and emulator	2
	1.1	Download and install ADT (Android Developer Tools)	2
	1.2	Prepare and control your emulator	3
	1.3	Monitor and debug in the emulator	6
2	Under	stand Android apps	7
	2.1	Application components	7
	2.2	Application resources	8
	2.3	App properties	9
3	Create	e your first Android app	10
4	Run ai	nd debug your apps	13
	4.1	Use an emulator	13
	4.2	Use a real device	14
5	Addin	g more UI elements	17
	5.1	Understand the UI XML layout structure	18
	5.2	Add an image to your app	19
6	Handl	e user input & using WebView for HTML5 mobile app	21
7.	Publis	h your app	28
	7.1.	Prepare your app icon	28
	7.2.	Change the app name and apply localization	29
	7.3.	Sign and export the APK	30
8.	Refere	ence and learning resources:	33

Prepared by Mr. Felix Tam, Committee Member of the Special Interest Group (SIG) on Innovative Software 2013-2014, Department of Computer Science, Hong Kong Baptist University.

All rights reserved. All content copyright and other rights reserved by its respective owners. Any content, trademark(s), or other material that may be found on this document remains the copyright of its respective owner(s). In no way does the Special Interest Group on Innovative Software claim ownership or responsibility for such items, and you should seek legal consent for any use of such materials from its owner.

## **1** Setup development environment and emulator

#### 1.1 Download and install ADT (Android Developer Tools)

Tools that you will need for Android app development:

Option 1:

• Eclipse IDE with built-in ADT (Android Developer Tools)

Option 2:

- Eclipse IDE for Java Developers
- Java SDK
- ADT plugin for Eclipse
- Android SDK

We will choose option 1 in this tutorial (64bit) and please get a copy of the **All-in-One** package from: <u>http://developer.android.com/sdk/index.html</u>

\*Please make sure you have your JDK 1.7 (x64) installed on your computer. You may get it from

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-18802 60.html.

Extract the zip file in C:\ and navigate to C:\adt-bundle-windows-x86\_64-20131030. Double click eclipse.exe to start the Android development tool. Choose your own workspace when asked.



#### 1.2 Prepare and control your emulator

The package contains the latest Android system image (4.4, as of 2013-12-10) and we will base on this to create an Android 4.4 emulator. Except where noted, examples used in our tutorials (Session 1 - 3) are still compatible with Android API Level 8 or above.

## Close the **Welcome!** tab and click on the icon of **Android Virtual Device Manager** in the toolbar.



Click on **New** and input the parameters as the screenshot shown below.

🚯 Create new Andro	id Virtual Device (AVD)						
AVD Name:	my_avd						
Device:	Nexus 4 (4.7", 768 × 1280: xhdpi) 👻						
Target:	Android 4.4 - API Level 19 🔹						
CPU/ABI:	ARM (armeabi-v7a)						
Keyboard:	Hardware keyboard present						
Skin:	Display a skin with hardware controls						
Front Camera:	None						
Back Camera:	None						
Memory Options:	Memory Options: RAM: 512 VM Heap: 64						
Internal Storage:	200 MiB -						
SD Card:	<ul> <li>● Size:</li> <li>● File:</li> <li>● Browse</li> </ul>						
Emulation Options: Snapshot Use Host GPU							
OK Cancel							

Click on Start to start the emulator. If you received an error message " Emulator error: This AVD's configuration is missing a kernel file", please make sure the system environment variable is set to the path of your SDK directory with variable name ANDROID\_SDK\_ROOT

To set the **system environment variable**, click on **Start > Right click Computer > Advanced system settings > Advanced > Environment Variables**. Click **New** or **Edit ANDROID\_SDK\_ROOT** variable, and then specify the path of the SDK folder. E.g. C:\ adt-bundle-windows-x86\sdk

Also, if you received an error message "PANIC: Could not open: my\_avd", please copy the "avd" folder from "C:\Users\test1\.android" to "C:\Users\your\_id\.android".

ndroid Virtual D	evices Device Definit	ions						
List of existing Android Virtual Devices located at C:\Users\felixtam\.android\avd								
AVD Name	Target Name	Platform	API Level	CPU/ABI	New			
✓ my_avd	Android 4.4	4.4	19	ARM (armeabi	Edit Delete Repair Details Start			
🖌 A valid Andı	roid Virtual Device. 🔜	A repairable Android \	/irtual Device.		Refresh			

After the Android system is started successfully, you will see the following screen in the emulator.

\*If the **Auto Monitor Logcat** dialog is promoted, select **Yes, monitor logcat...** and click the **OK** button.



A list of shortcuts which are useful for controlling the emulator during testing

Emulated Device Key	Keyboard Key
Home	HOME
Menu	F2 or Page-up button
Back	ESC
Call/dial button	F3
Hangup/end call button	F4
Search	F5
Power button	F7
Switch to previous layout orientation	
(for example, portrait, landscape)	KEYPAD_7, Ctrl-F11
Switch to next layout orientation	
(for example, portrait, landscape)	KEYPAD_9, Ctrl-F12
Toggle cell networking on/off	F8
Toggle full screen mode	Alt-Enter
Toggle trackball mode	F6
Enter trackball mode temporarily	
(while key is pressed)	Delete

5

#### 1.3 Monitor and debug in the emulator

Click on **Window > Open Perspective > DDMS** to open DDMS (Dalvik Debug Monitor Server) perspective. Below are some of the functions you can do with DDMS:

- View files inside the emulator
- Simulate different telephony statuses
- Simulate the longitude and Latitude value
- View the system message including the exceptions thrown by your app using LogCat.

ODMS - ADT					- • ×
<u>File Edit N</u> avigate Se <u>a</u> r	rch <u>P</u> roject <u>I</u>	Run <u>W</u> indow	v <u>H</u> elp		
📑 • 🛛 🖷 📥 💁 •	🔗 • 🗄 •	§ • %	$\leftarrow \bullet \bullet \bullet   \ge$	Quick Access 📔 🛍 .	lava 👩 DDMS
Devices 🔀			🔆 Threads 🔒 Heap 🔒 Allocation 🗢 Network St 🍎 F	File Explorer 💿 Emulator 🐹 🗖 System	Inf 🗖 🗗
*	•	(B) (8)			
	s   •••   ••••		The loss of the		
, i i i i i i i i i i i i i i i i i i i			l elephony Status		
Name			Voice: home		
🔺 💷 my_avd [emulat O	nline	my_avd [	Data: home  Iatency: None		
android.proc 79	91	8600			
com.androic 68	35	8601	Telephony Actions		
system_proc 36	59	8602	Incoming number:		
com.androic 11	130	8603	@ Voice		
com.androic 70	)4	8604	() voice		-
android.proc 55	55	8605	© SMS		=
com.androic 91	15	8606	Message:		A
com.androic 43	38	8607			
com.androic 10	014	8608			
com.androic 59	33	8609			-
com.androic 10	148	8610			
com.androic b2	25	8611	Call Liang op		
com.androic /s	34	8012	Location Controls		
com.androic /a	50	8013	Manual CDV LIGH		
com.androic 5/	10 De	0014	Manual GPX KML		
com.androic 63	50 NG	8015	O Decimal		
com.androic ou	70	0010	Sexagesimal		
com.androic 12		0017	Longitude -122.084095		
			Latitude 37.422006		
•		•	Send		-
	_			2019 Strate Bootherson	
🗐 LogCat  📃 Console 🖇	3			🗟 🔂 🛃 🚽 🖛	1
DDMS					
					-
4					Þ
			,		
			40M of 241	M	

## 2 Understand Android apps

#### 2.1 Application components

#### Activities

- Each activity is a single screen with a user interface
- Every screen is implemented as a subclass of android.app.Activity



#### Services

- Run in background
- No user interface
- Designed for performing tasks running in background without user interaction

Every service is implemented as a subclass of android.app.Service



#### Fragment (Introduced since API Level 11 / v4 support library)

- Represents a particular interface which encapsulated in an activity.
- Reusable UI
- Excellent interaction with ActionBar

		11:49	
НКВИ		:	
INBOX	)ffice - HKBU	Dec 19	
Inbox	dvancement Office, HKBU — http:// 1219-1.html		
Priority Inbox		Inbox	
	013 — If you can't view this email, please Spring Gathering (15		$\bigtriangleup$
RECENT LABELS		Inbox	
НКВИ	Office - HKBU	Dec 18	
		₩ <b>4</b>	Ś
ALL LABELS		Inbox	
Starred	essare-Id:	Dec 18	
otaricu	ildist02 hkbu edu bk> Date: Wed, 18 Dec	55	

#### **Content providers**

- A shared set of application data
- E.g. Contacts, Calendar (Android 4.0)

#### **Broadcast receivers**

- A component designed to respond to global broadcast announcements
- System broadcast announcements E.g. change of battery status/network status etc.

#### **2.2 Application resources**

- Provide resources to your apps. E.g. Image, Video, Audio, Text etc...
- Resource are put under **res** directory of your Android project
- Every resource is identified by a resource ID
- Resources can have qualifier names for alternatives E.g. layout-land, values-ja, drawable-hdpi etc

SIG on Innovative Software, COMP, HKBU

2013 Winter Workshop on Android App Development Session 1



#### 2.3 App properties

We use a file called AndroidManifest.xml to describe the information about your

app to the system and Google play as well. Inside this file, you can:

- Define your package name
- Define your version code and version name
- Define your app icon
- Define your App components (Activities/Services/Receivers etc.)
- Define the permission(s) should be granted



#### You may refer to

http://developer.android.com/guide/topics/manifest/manifest-intro.html for details.

## 3 Create your first Android app

1.	Start Eclipse,	select File :	> New >	Android	Application	Project.

		. IA		
New	Alt+Shift+N ►		Java Project	
Open File			Android Application Project	
Close	Ctrl+W	Ľ	Project	
Close All	Ctrl+Shift+W	₿	Package	
Save	Ctrl+S	C	Class	
Save As		Ø	Interface	
a Save All	Ctrl+Shift+S	G	Enum	
Revert		@	Annotation	
Maya		<b>₽</b> ₽	Source Folder	
Pename	E2	Ø	Java Working Set	
Befrech	F2 F5		Folder	
Convert Line Delimiters To		Ľ	File	
convert Line Delimiters To	•		Untitled Text File	
Print	Ctrl+P	ď	Android XML File	
Switch Workspace		E	JUnit Test Case	
Restart		C2	Example	
Import		<b>1</b>	Other	Ctrl+N
🚹 Export				
Properties	Alt+Enter			
1 dump_5397806062556393611.uix [U	1 dump_5397806062556393611.uix [Users]			
Exit				

 Input the required information as below. We will create an app with a blank activity equipped with ActionBar. Since we have checked the option "Create Activity" during project setup, the ADT will automatically generate a default activity and related resources right after the project is created.

() New Android Applicatio	n	
New Android Application	on le.' is meant as a placeholder and should not be used	0
Application Name: Project Name: Package Name: &	HelloWorld HelloWorld com.example.helloworld	
Minimum Required SDK:0	API8: Android 2.2 (Froyo)	
Compile With 0	API15: Android 4.4 (KitKat)	
Theme:	Holo Dark	
ineme:o		
Choose the base ther	ne to use for the application	
?	< <u>B</u> ack <u>N</u> ext >	Einish Cancel
New Android Applicatio	n	_ <b>0 X</b>
New Android Application Configure Project	pn r icon	0
Mark this project as a li	brary	
Create Project in Works	pace	
Location: C:\adt-bundle	-windows-x86_64-20131030\eclipse\workspace\HelloW	orld Browse
Working sets	ng sets	
Working sets:		▼ S <u>e</u> lect
?	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish Cancel

() New Android Application	
Create Activity Select whether to create an activity, and if so, what kind of activity.	0
Create Activity BlankActivity FullscreenActivity LoginActivity MasterDetailFlow SettingsActivity	(
New Blank Activity Creates a new blank activity, with optional inner navigation.	
? < <u>B</u> ack Next >	<u>Finish</u> Cancel

🕖 New Android Ap	oplication	
New Blank Activ	<b>ity</b> nk activity, with optional inner navigation.	0
		(
Activity Name®	MainActivity	
Layout Name®	activity_main	
Navigation Type®	None	
The type of navi	gation to use for the activity	
?	< <u>B</u> ack <u>N</u> ext >	<u>Finish</u> Cancel

That's it. You have your first **HelloWorld** Android app.

## 4 Run and debug your apps

#### 4.1 Use an emulator

To run your app in an emulator, right click on the project in Eclipse and choose Run

#### As > Android Application.



Choose the emulator (if more than one emulator or device attached) you are going

to use	2.								
🕖 And	Iroid Device Choo:	ser							<b>—</b> ×
Select Cho	a device with min a	API level 8. droid device							
	Serial Number		AVD Name		Ta	rget		Debug	State
	💷 my_avd [emu	lator-5554]	my_avd		<ul> <li>Android 4.4</li> </ul>		Yes	Online	
	冒 lge-nexus_5-034d16c1094b2ae2		N/A		<ul> <li>✓ 4.4</li> </ul>			Online	
🔘 Lau	nch a new Android	Virtual Device							
	AVD Name	Target Name		Platform	API	_evel	CPU/A	BI	Details
		No AVD availab	le						Start

After a while the emulator should start your app



#### 4.2 Use a real device

Make sure the driver of your device is properly installed in **Device Manager** otherwise the ADB will not be able to detect the present of your device. You may visit <u>http://developer.android.com/tools/extras/oem-usb.html#Drivers</u> for the driver details of your device.



On your device, go to **System settings > Developer options** and enables the **USB debugging** option.



Run the application from Eclipse using the same steps written for an emulator and select your real device in **Android Device Chooser** window.

	হ 🕼 🗹 15:14
蘭 HelloWorld	:
Hello world!	
Ú Ú	

The result should be as same as using the emulator.

## **5** Adding more UI elements

All UI elements are built using View and ViewGroup objects in an application. The View can be extended to a TextView, ImageView, WebView etc, and the ViewGroup serves as layout architecture which describes how the View to be organized in an activity. Apart from building UI directly using Java code, we can



also create the interface using XML document. By using XML markups, we can simplify our code and make debugging easier.

The XML layout file is saved in **res/layout**. By using the HelloWorld app as an example, you will find the initial UI is defined in activity\_main.xml. When you open the activity\_main.xml, Eclipse will show you a graphical UI editor for the interface preview.

🕡 Java - HelloWorld/res/layout/activity_main.xml - ADT						
<u>File Edit Navigate Search Project Refactor Run</u>	<u>W</u> indow <u>H</u> elp					
* • • • • • • • • • • • • • • • • • •	<b>2</b> •   ×   ⊕ ♂ •   <i>A</i> • (	∋[≱ - ₽ - + + -	• + •   e		Quick	Access 🛛 🔡 📳 🐉 Java 🚳 DDMS
😫 Package Explorer 💥 📄 🖨 🗖	activity_main.xml ☆					🗖 🔠 Outline 🛛 🗖 🗖
<sup>[2]</sup> HelloWorld <sup>[2]</sup> Gen [Generated Java Files] <sup>[2]</sup> gen [Generated Java Files]	Palette     Palette     Palette     Form Widgets	▼	sOne ▼ 🛛 🗗	👻 🛧 AppTheme 💌	MainActivity	RelativeLayout     AD TextView - "Hello world!"
<ul> <li>Android Private Libraries</li> <li>Android Private Libraries</li> <li>assets</li> <li>bin</li> <li>bin</li> </ul>	TextView Large Medium Small Butter Small OFF ✓ CheckBox ● RadioButton CheckedTextView		•   EE	<b>V</b> 8	@.@.0. 0.€	2
Byres     Constant of the second	Spinner Subitem	Hello world!	World	ы		
Permenu     Permenu     Permenu     Permenu     Permenues-swooddep     Permenues-swooddep     Permenues-swooddep     Permenues-swooddep     Permenues-swooddep     Permenues-swooddep     Permenues-swooddep     Permenues-swooddep						Propertie 🎲 🖧 📖 🗈 🗖 Id un A Backgrou Padding L. @dimen/activity m
<ul> <li>Andowania Extant</li> <li>Launcher-web.png</li> <li>proguard-project.txt</li> <li>project.properties</li> </ul>	Layouts     Composite			- 84		Content
	Time & Date			- 15		View [] Style
	Transitions	_		- 15		Tag 🔤 Backgr
	Advanced     Other			_		Padding
	🗀 Custom & Library Views	*			Þ	Paddin @dimen/activity
	Graphical Layout 🕞 activ	rity_main.xml				Paddin @dimen/activity
	📳 Problems @ Javadoc 😡	Declaration 😰 LogCat 😒				- 8
	Saved Filters 🕂 🗕 📓	earch for messages. Accepts	Java regexes. Pre	fix with pid:, app:, tag: or te	d: to limit scope.	verbose 🔻 🖬 🖳 🛓
	All messages (no filters)	Time	PID TID	Application	Tag T	ext
	com.example.hellowori D D D	07-05 10:44:3 07-05 10:44:3 07-05 10:44:3 07-05 10:44:3	7576 7576 7576 7576 7576 7576 7576 7576	<pre>com.example.hel com.example.hel com.example.hel com.example.hel</pre>	libEGL 1 libEGL 1 libEGL 1 OpenGLRend E	oaded /vendor/lib/egl/libEGL_FOWERVR oaded /vendor/lib/egl/libGLESv1_CM_F oaded /vendor/lib/egl/libGLESv2_FOWE nabling debug mode 0
	•			m		•
				1	10M of 238M	

However, the GUI editor is not sophisticated enough. The final appearance of the interface layout may differ from the preview in the editor. Therefore, the best practice is to create the UI elements by using XML markups directly. To do this, you can click on the **xxx.xml** tab.

#### 5.1 Understand the UI XML layout structure

Using the activity\_main.xml as an example:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools="http://schemas.android.com/tools" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:paddingBottom="@dimen/activity\_vertical\_margin" android:paddingLeft="@dimen/activity\_horizontal\_margin" android:paddingRight="@dimen/activity\_horizontal\_margin" android:paddingTop="@dimen/activity\_vertical\_margin" tools:context=".MainActivity" >

<TextView

android:layout\_width="wrap\_content"
android:layout\_height="wrap\_content"
android:text="@string/hello\_world" />

#### </RelativeLayout>

The **RelativeLayout** is a **ViewGroup** which holds its children in a relative way. The above **RelativeLayout** has only one child – **TextView**. Android uses **TextView** to display text to users. For a complete list of **ViewGroups** and **Views**, please refer to <u>http://developer.android.com/reference/android/view/ViewGroup.html</u> and <u>http://developer.android.com/reference/android/view/View.html</u> respectively.

Each kind of **ViewGroups** or **Views** has its own attributes. The attributes usually start with **android:xxx** where the **xxx** is the property name of an attribute. Different **ViewGroups** and **Views** share some common attributes but some of them are unique.

To have a quick outlook on different layouts and UI elements, please go to <a href="http://developer.android.com/guide/topics/ui/declaring-layout.html#CommonLayou">http://developer.android.com/guide/topics/ui/declaring-layout.html#CommonLayou</a> <a href="http://topics/ui/declaring-layout.html#CommonLayou">ts for the samples. Now, we will see how to add an image just below the text in our activity\_main.xml.">activity\_main.xml</a>

#### 5.2 Add an image to your app

- Prepare an image. You may download this image <u>http://developer.android.com/images/sdk-cube.png</u> as an example.
- 2. Put the image under **res/drawable-hdpi**. If this folder is not found, you can create it manually using the **Package Explorer** in Eclipse.
- Drag the image directly to the drawable-hdpi folder through the Package
   Explorer. In the File Operation, choose Copy files. You also need to rename the file if the characters are not in (a-z), (0-9), (\_) and (.)



- In the res/layout/activity\_main.xml, edit the TextView attribute
   Add: android:id="@+id/textView1"
- Add the following XML markup below the **TextView** <ImageView</li>

android:layout\_width="wrap\_content"
android:layout\_height="wrap\_content"
android:src="@drawable/sdk\_cube"
android:layout\_centerHorizontal="true"
android:layout\_below="@id/textView1"/>

Code explanation:

- The ImageView is used to show an image in the interface.
- The android:src is used to specify the source of the image file. Here the path and the file name are @drawable/sdk\_cube. (File extension is omitted)
- The android:layout\_width and android:layout\_height="wrap\_content" is used to specify the ImageView size which is the same as the image itself.
- 6. You can press the **Graphical Layout** tab to preview the result.

7. Run your app in the emulator and see the actual result. The final appearance should look like this:



Complete project and sample codes available at HelloWorld.zip

# 6 Handle user input & using WebView for HTML5 mobile app

In this section, we will create an app to intercept user input and incorporate some HTML5 mobile app frameworks. Below are some frameworks which are popular:

- Sencha Touch
- jQuery Mobile
- jQTouch
- 1. Create a new Android project and input the required information as shown below.

🕠 New Android Applicatio	n		
New Android Application	on e.' is meant as a placeholder and should not be used		
Application Name:	HelloHtml5		
Project Name:	HelloHtml5		
Package Name: 🌢	com.example.hellohtml5		
Minimum Required SDK:	API 8: Android 2.2 (Froyo)		
Target SDK:0	API 19: Android 4.4 (KitKat)	<b>•</b>	
Compile With:	API 19: Android 4.4 (KitKat)	•	
Theme:0	Holo Dark	•	
• Choose the base theme to use for the application			
?	< Back Next >	<u>F</u> inish Cancel	

2. Click Next three times and choose BlankActivity.

O New Android Application	
Create Activity Select whether to create an activity, and if so, what kind of activity.	<b>(</b>
Select whether to create an activity, and if so, what kind of activity.  Create Activity BlankActivity FullscreenActivity MasterDetailFlow SettingsActivity New Blank Activity	
Creates a new blank activity, with optional inner navigation.	
(?) < <u>Back</u> <u>Next</u> >	<u>Finish</u> Cancel

- 3. Click **Next** and **Finish** to create the project.
- 4. Delete the default generated **TextView** in activity\_main.xml
- 5. Remove the attributes android:paddingXXXX from the RelativeLayout.
- Add a LinearLayout inside the default RelativeLayout in activity\_main.xml using the markup below. We will use this LinearLayout to hold the address bar (EditText) and the Go button.

<LinearLayout

android:layout\_width="match\_parent"
android:layout\_height="wrap\_content"
android:id="@+id/linearLayout1">

</LinearLayout>

7. Inside this LinearLayout, we will add an EditText as an address bar and add a button next to it. To add an EditText, use the markup below:

<EditText

android:id="@+id/editText1"
android:layout\_width="wrap\_content"
android:layout\_height="wrap\_content"
android:layout\_weight="1"
android:hint="Enter URL"

22

#### android:ems="10"/>

8. To add a button, use the markup below:

<Button

android:id="@+id/button1"
android:layout\_width="wrap\_content"
android:layout\_height="wrap\_content"
android:layout\_weight="0"
android:text="Go"/>

 In this step we will add a WebView under the address bar (responsible for loading and showing the HTML5 mobile app framework). Add the following markup to res/layout/activity\_main.xml below the LinearLayout created in step 5.

<WebView

android:id="@+id/webView1"
android:layout\_width="match\_parent"
android:layout\_height="match\_parent"
android:layout\_below="@id/linearLayout1"/>

10. Now try to run the app and it should look like this:



 Starting from this step we will focus on the Java code. Open MainActivity.java under src > com.example.hellohtml5.



12. Create three private variables above the **onCreate** method, import necessary packages using Eclipse:

private Button button; private EditText editText; private WebView webView;

13. Inside the **onCreate** method, add the following codes after

setContentView(R.layout.activity\_main):

button = (Button)findViewById(R.id.button1); editText = (EditText)findViewById(R.id.editText1); webView = (WebView)findViewById(R.id.webView1); webView.getSettings().setJavaScriptEnabled(true); webView.setWebViewClient(new WebViewClient());

\*import android.webkit.WebView; android.webkit.WebViewClient; android.widget.Button and android.widget.EditText; to fix errors due to missing classes.

- 14. Create an **OnClickListener** for the button inside the MainActivity class. This listener will achieve the following goals:
  - 1. Intercept the click action of the button
  - 2. Read the URL typed in the EditText
  - 3. Instruct the WebView to load the page according the URL

private OnClickListener buttonClick = new OnClickListener(){
 @Override
 public void onClick(View arg0) {
 webView.loadUrl(editText.getText().toString());
 }
};

\*You should import **android.view.View.OnClickListener** when prompted for **OnClickListener** in Eclipse. Import **android.view.View** too.

- 15. Bound the **OnClickListener** to the Go button using follow code in onCreate(): button.setOnClickListener(buttonClick);
- 16. Since the app will access Internet to load the HTML5 framework demo, we need to declare necessary permission in AndroidManifest.xml. Add the following permission declaration above the application element. <uses-permission android:name="android.permission.INTERNET" />
- 17. Run the app and test the HTML5 frameworks, type the links below in the address bar and tap on the **Go** button.
  \*You are recommended to use a real device instead of emulator for having a better experience in this example.
  Sencha Touch

Full URL: <u>http://cdn.sencha.io/touch/sencha-touch-2.1.0/examples/kitchensink/inde</u> <u>x.html</u> Short URL: <u>http://goo.gl/CLm5zE</u>



#### jQuery Mobile

Full URL: <u>http://demos.jquerymobile.com/1.4.0/</u> Short URL: <u>http://goo.gl/yYjJOr</u>





#### jQTouch ullet

Full URL: http://jqtouch.com/preview/demos/main/#home Short URL: http://goo.gl/M3LK0

	❤∕ 🛿 4:34
🚔 HelloHtml5	:
http://goo.gl/M3LK0	Go
jQTouch	About
User Interface	5 >
Animations	7 >
AJAX	4 >
Events	4 >
Extensions	4 >
Demos	2 >
Themes	2 >
External Links	
Homepage	0
Û Û	

Complete project and sample codes available at HelloHtml5.zip

## 7. Publish your app

Before publishing the app to Google Play Store or distribute it to someone else, you should design an icon for your app instead of using the default one. Also, you may need to change the app name according to your preference later on.

#### 7.1. Prepare your app icon

For the app icon, we have to prepare at least four different sizes for different screen resolutions. Namely: ldpi, mdpi, hdpi and xhdpi (There is also tvdpi). In the following steps, we will base on the previous example which is completed in section 6.

 Please prepare an app icon and resize it to 36x36px for ldpi, 48x48 for mdpi, 72x72 for hdpi and 96 x 96 for xhdpi. For the app icon design guideline. Please refer to

http://developer.android.com/guide/practices/ui\_guidelines/icon\_design\_lau ncher.html. You can also get some free icons at <a href="http://www.iconarchive.com">www.iconarchive.com</a>

- 2. Put the icon with the same file name (E.g. icon.png) into drawable-hdpi (72x72), drawable-mdpi (48x48) and drawable-ldpi (36x36) respectively.
- Open the AndroidManifest.xml, change android:icon="@drawable/ic\_launcher" to android:icon="@drawable/icon". You can delete the default launcher icon afterwards.



The new icon will be shown in the launcher after you deployed your app into the emulator or device.



#### 7.2. Change the app name and apply localization

When you look at the **AndroidManifest.xml** file, you can see an attribute android:label="@string/app\_name" is already added to the <application> tag. The @string/app\_name refers to a value of an elements call app\_name located in value/strings.xml. By storing the string in the values/strings.xml, we can easily localize our apps by using different qualifier name (language code this time). E.g.: it for Italian, zh-rTW for Traditional Chinese, and zh-rCN for simplified Chinese

- 1. Go to values/strings.xml, change the app name to CSD.
- 2. Create a folder called values-it in res directory.
- Create a strings.xml file inside the values-it and insert the following markup: <?xml version="1.0" encoding="utf-8"?> <resources>

<string name="app\_name">Dipartimento di Informatica</string> </resources>

- 4. Run your app in the emulator or device, you can see the app name is changed.
- Now, change the locale setting in the emulator or device by going to Settings > Language and input > Language > Italiano. You should see the name of the app



will be automatically changed to Italian.

#### 7.3. Sign and export the APK

For every Android app, you have to sign it before uploading it to Google Play Store or distributing it to others. The certificate used by ADT for deploying your app in the emulator or device is a debug key only. When exporting the Android app in Eclipse, you can create your own private key if you have not yet created.

 In Eclipse, right click your project and select File > Export > Android > Export Android Application.

O Export	
Select	2
Select an export destination:	
type filter text	
<ul> <li>▷ Ceneral</li> <li>▷ Android</li> <li>▷ Export Android Application</li> <li>▷ ▷ C/C++</li> <li>▷ ▷ Install</li> <li>▷ ▷ Java</li> <li>▷ ▷ Team</li> <li>▷ ▷ XML</li> </ul>	
	sh Cancel

- 2. Click **Next** to continue if the selected project is correct.
- 3. Select **Create new keystore**, choose the location for saving the new keystore and enter the password.

🚺 Export A	ndroid Application
Keystore s	election
<ul> <li>Use exis</li> <li>Create r</li> </ul>	ting keystore new keystore
Location:	C:\Program Files (x86)\adt-bundle-windows-x86\eclipse\mykey Browse
Password:	•••••
Confirm:	•••••
?	< <u>Back</u> <u>Next</u> <u>Finish</u> Cancel

4. Enter your personal information in **Key Creation** step. Below is a sample screenshot:

() Export Android Ap	plication
Key Creation	<u></u>
Alias:	androidworkshop
Password:	•••••
Confirm:	•••••
Validity (years):	100
First and Last Name:	Felix Tam
Organizational Unit:	Department of Computer Science
Organization:	Hong Kong Baptist University
City or Locality:	Hong Kong
State or Province:	China
Country Code (XX):	
?	< <u>Back</u> <u>N</u> ext > <u>Finish</u> Cancel

5. Choose your location for saving the APK file.

Export Android Apple 1	oplication	×
Destination and k	ey/certificate checks	
Destination APK file:	C:\Program Files (x86)\adt-bundle-windows-x86\eclipse Browse	
Certificate expires in	100 years.	]
?	< <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel	

6. Now you are ready to distribute your APK file to Google Play Store or install it directly in any devices.

## 8. Reference and learning resources:

- Official Android developer website: http://developer.android.com/index.html
- Many common questions raised by other developers previously: <u>http://stackoverflow.com/</u>
- Many Android tutorials with complete source code: <u>http://www.anddev.org</u>
- Different ways for implementing listeners in Android app <u>http://tseng-blog.nge-web.net/blog/2009/02/14/implementing-listeners-in-you</u> <u>r-android-java-application/</u>
- Special Interest Group on Innovative Software homepage: http://www.comp.hkbu.edu.hk/~sigis/