ANDROID RESOURCE AND MENU

Nam Nguyen Tu
Linh Ha Nguyen

23-6-2013

Introduction

- Your role
- Your background and experience in the subject
- What do you want from this course

Course Objectives

- At the end of the course, you will have acquired sufficient knowledge to:
  - Understand Android resource
  - Understand Android menu
  - Apply them in exercises

Agenda

Course Audience and Prerequisite

- The course is for programmers which interest in Android development.
- The following are prerequisites to this course:
  - Java and OO knowledge
  - XML knowledge
  - MVC/MVP or MVVM knowledge
  - Android Overview and Basic

Assessment Disciplines

- Class Participation: 40%
- Assignment: 60%
- Final Exam: 0%
- Passing Scores: 70%

Further References

- http://developer.android.com/guide
- Wikipedia

Set Up Environment

- To complete the course, your PC must install:
  - Android SDK
  - Eclipse with ADT plugin

Course Administration

- In order to complete the course you must:
  - Sign in the Class Attendance List
  - Participate in the course
  - Provide your feedback in the End of Course Evaluation

ANDROID RESOURCES

Introduction about Android resources and how to apply it

OVERVIEW

- What is resource in Android ?
  - Images

  - Strings

"Congrat ! You have completed ..

  - Other special files…

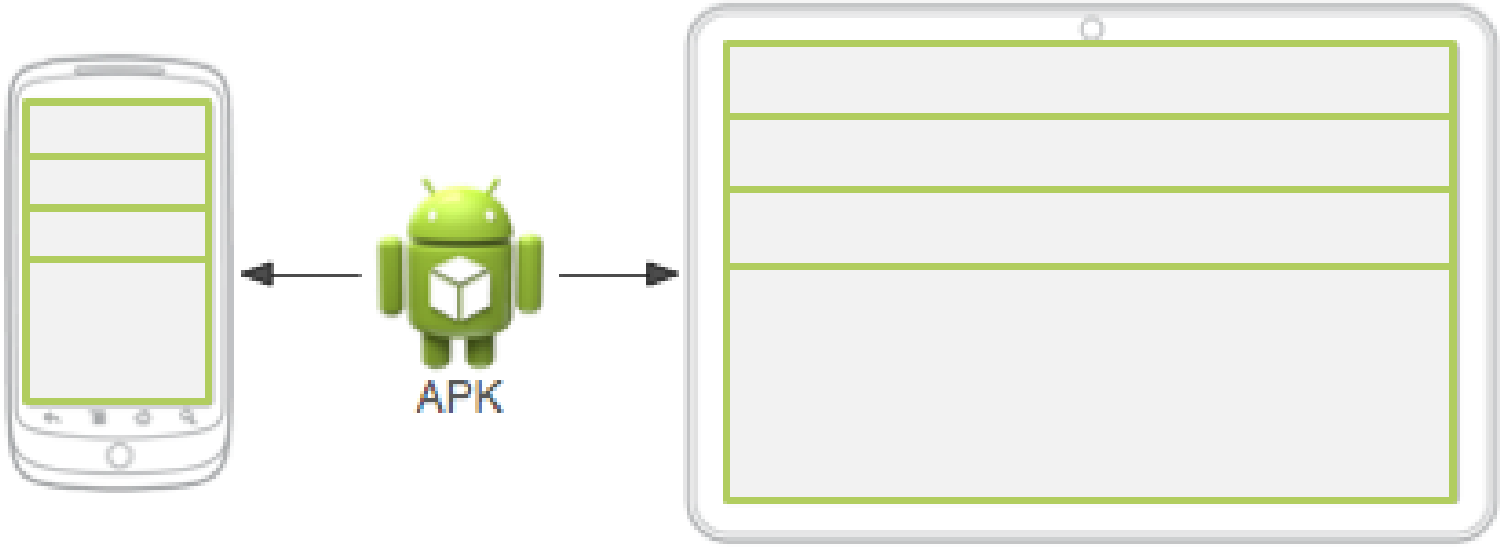
Sound, mp3 …

OVERVIEW

- Your app run on many devices
- They have different resolution
- How to let the application run on every devices without writing new apk for each new device.

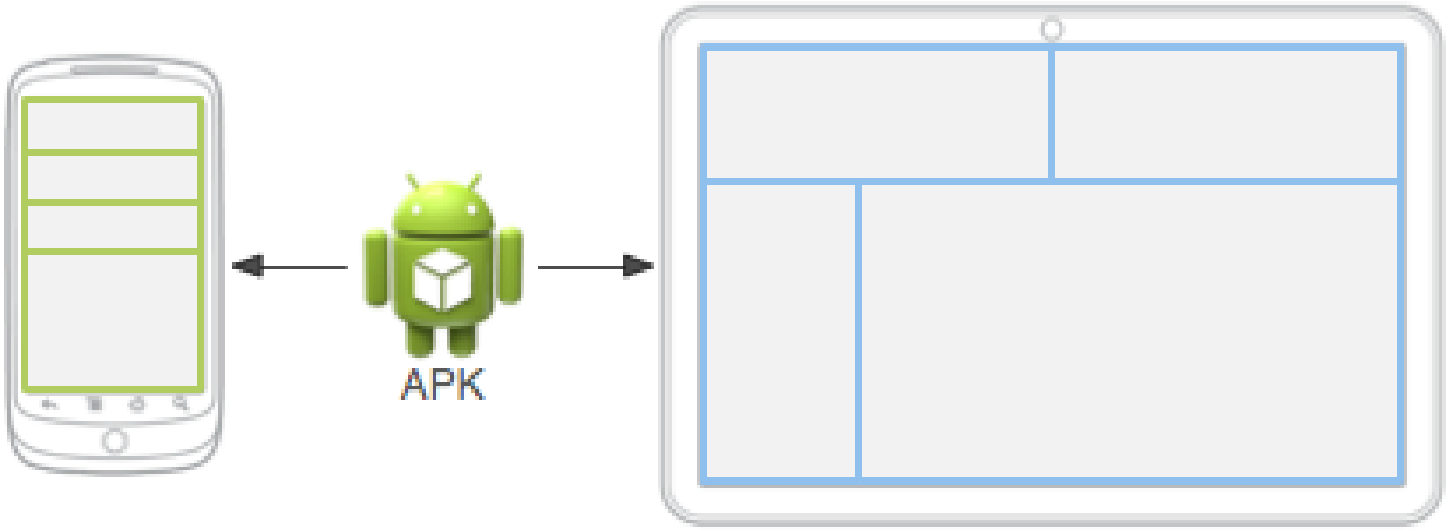
DEFAULT RESOURCES

- Should be used regardless of the device configuration
- When there are no “matched” resources that match the current configuration.



ALTERNATIVE RESOURCES

- Designed for use with a specific configuration.
- It is appended an appropriate configuration qualifier to the directory name.



ALTERNATIVE RESOURCES

- VERTICAL:
  - Default is saved in res\layout
- HORIZONTAL:
  - Alternative is saved in res\layout-land

- Android automatically applies the appropriate resources by matching the device's current configuration to your resource directory names.
- It means that we just define folder for many configurations and put resources in it, Android will apply it for you.
- No need to check in code
- It’s too convenience.

STEP APPLY RESOURCES

- Providing Resources
  - What kinds of resources you can provide in your app
  - where to save them
  - how to create alternative resources for specific device configurations.
- Accessing Resources
  - How to use the resources you've provided
    - from your application code
    - from other XML resources.
- Handling Runtime Changes
  - How to manage configuration changes that occur while your Activity is running.

PROVIDING RESOURCES

- each type of resource in a specific subdirectory of **res/**

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            icon.png
        layout/
            main.xml
            info.xml
        values/
            strings.xml
```

- the **res/** directory contains all the resources (in subdirectories):
  - image resources
  - layout resources
  - string resource files.
- The resource directory names are important

PROVIDING RESOURCES

- Never save resource files directly inside the res/ directory—it will cause a compiler error.
- It should stay in sub directories of res/

animator/	XML files that define property animations.
anim/	XML files that define tween animations.
color/	XML files that define a state list of colors.
drawable	Bitmap files (.png, .9.png, .jpg, .gif)
layout/	XML files that define a user interface layout
menu/	XML files that define application menus
raw/	Arbitrary files to save in their raw form
values	XML files that contain simple values, such as strings, integers, and colors
xml/	Arbitrary XML files that can be read at runtime by calling Resources.getXML()

PROVIDING RESOURCES

- Almost every application should provide alternative resources to support specific device configurations.
- For instance, you should include
  - alternative drawable resources for different screen densities
  - alternative string resources for different languages.
- At runtime, Android detects the current device configuration and loads the appropriate resources for your application.
- Configuration-specific alternatives for a set of resources
  - in res/ named in the form **<resources_name>-<config_qualifier>**

PROVIDING RESOURCES

- Many kinds of resources need to provide in many cases.
- Android apply resources base on qualifier
- Screen size
  - small : QVGA screen
  - normal : HVGA screen
- UI mode
  - car : in a car dock.
  - television: device is played on TV.
- Screen orientation
  - Port : Portrait
  - Land: Landscape
- And more …..

ACCESSING RESOURCES

- Apply it by referencing its resource ID
- All resource IDs are defined in your project's R class, which the aapt tool automatically generates.
- For each type of resource, there is an R subclass
- Syntax to reference a resource in code:

[<package_name>.]R.<resource_type>.<resource_name>

ACCESSING RESOURCES

- Use in code:
- `getWindow().setBackgroundDrawableResource(R.drawable.my_background_image) ;`
- `getWindow().setTitle(getResources().getText(R.string.main_title));`
- `setContentView(R.layout.main_screen);`

ACCESSING RESOURCES

- Can use resource in XML

```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textColor="@color/opaque_red"
    android:text="@string/hello" />
```

- In XML do not need to use R.string.hello
- It is replaced by @string/hello.

ACCESSING RESOURCES

- Android contains a number of standard resources, such as styles, themes, and layouts.
- To access these resource, qualify your resource reference with the android package name.

```
setListAdapter(new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, myarray));
```

- In this example, simple_list_item_1 is a layout resource defined by the platform for items in a ListView.

HANDLING RUNTIME CHANGES

- Some device configurations can change during runtime:
  - Screen orientation
  - Keyboard availability
  - Language.

- In those changes, Android activities is stop and restart , which may can causes slow and lag.
- We should save the state of our activity while changing configurations.
  - Retain an object during a configuration change
  - Handle the configuration change yourself

HANDLING RUNTIME CHANGES

- Retain an object during a configuration change
  - Allow your activity to restart when a configuration changes, but carry a stateful Object to the new instance of your activity.

- Step to retain:
- Override the onRetainNonConfigurationInstance() method to return the object you would like to retain.
- When your activity is created again, call getLastNonConfigurationInstance() to recover your object.

HANDLING RUNTIME CHANGES

- Handle the configuration change yourself
  - Prevent the system from restarting your activity during certain configuration changes, but receive a callback when the configurations do change, so that you can manually update your activity as necessary.
- Define in XML which will be changed:

```
<activity android:name=".MyActivity"
        android:configChanges="orientation|keyboardHidden"
        android:label="@string/app_name">
```

- MyActivity will receive a callback onConfigurationChanged

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    // Checks the orientation of the screen
    if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
        Toast.makeText(this, "landscape", Toast.LENGTH_SHORT).show(); } }
```

LOCALIZATION

- The application's default text in a file with the following location and name:
  - res/values/strings.xml
- Contains English text for all the strings that the application uses
- Define other language in res/values-<<qualifier>>
- res/values-fr/strings.xml
  - Contain French text for all the strings
- res/values-ja/strings.xml
  - Contain Japanese text for all the strings

ANDROID MENU

Create Android menu

OVERVIEW

- Menus are a common user interface component in many types of applications.
- Beginning with Android 3.0 (API level 11), Android-powered devices are no longer required to provide a dedicated Menu button.
- Traditional 6-item menu panel < == > action bar.

MENU TYPE

- Options menu and action bar
  - It's where you should place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings."
- Context menu and contextual action mode
  - A floating menu that appears when the user performs a long-click on an element.
  - It provides actions that affect the selected content or context frame.
- Popup menu
  - A popup menu displays a list of items in a vertical list that's anchored to the view that invoked the menu.

CREATE MENU IN XML

- Menu and all its items are defined in an XML menu resource.

- Using a menu resource is a good practice for a few reasons:
  - It's easier to visualize the menu structure in XML.
  - It separates the content for the menu from your application's behavioral code.
  - It allows you to create alternative menu configurations for different platform versions, screen sizes, and other configurations by leveraging the app resources framework.

CREATE MENU IN XML

- **<menu>**
  - Defines a Menu, which is a container for menu items. A <menu> element must be the root node for the file and can hold one or more <item> and <group> elements.

- **<item>**
  - Creates a MenuItem, which represents a single item in a menu. This element may contain a nested <menu> element in order to create a submenu.

- **<group>**
  - An optional, invisible container for <item> elements. It allows you to categorize menu items so they share properties such as active state and visibility.

CREATE MENU IN XML

```
<?xml version="1.0" encoding="utf-8"?>
    <menu
xmlns:android="http://schemas.android.com/apk/res/android"
>
        <item android:id="@+id/new_game"
              android:icon="@drawable/ic_new_game"
              android:title="@string/new_game"
              android:showAsAction="ifRoom"/>
        <item android:id="@+id/help"
              android:icon="@drawable/ic_help"
              android:title="@string/help" />
    </menu>
```

CREATE MENU IN XML

- The <item> in the above menu has some important attributes:
- android:id
  - A resource ID that's unique to the item, which allows the application can recognize the item when the user selects it.
- android:icon
  - A reference to a drawable to use as the item's icon.
- android:title
  - A reference to a string to use as the item's title.
- android:showAsAction
  - Specifies when and how this item should appear as an action item in the action bar.

CREATE OPTION MENU AND ACTION BAR

- To specify the options menu for an activity, override onCreateOptionsMenu()
-
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```
-

CREATE OPTION MENU AND ACTION BAR

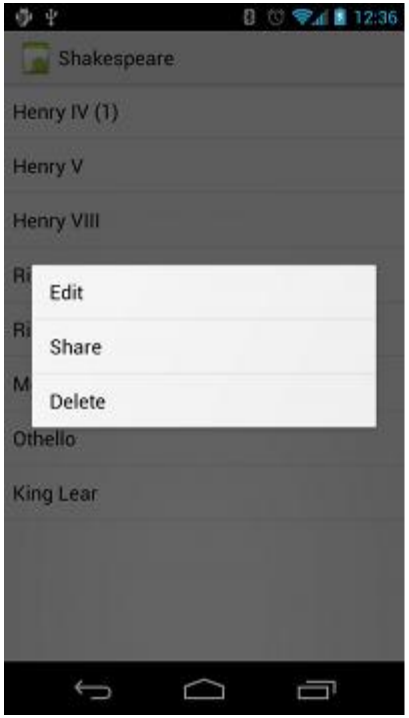
Android 2.3 and sooner	Android 3.0 and later
contents of your options menu appear at the bottom of the screen	items from the options menu are available in the action bar

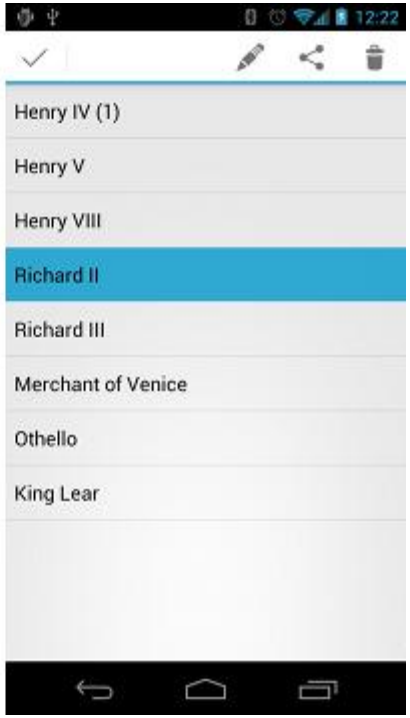
CREATE OPTION MENU AND ACTION BAR

- When the user selects an item from the options menu, the system calls your activity's onOptionsItemSelected() method
-
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
switch (item.getItemId()) {
        case R.id.new_game:
            newGame();
            return true;
        case R.id.help:
            showHelp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

CREATE CONTEXT MENU

- You can provide a context menu for any view, but they are most often used for items in:
  - ListView
  - GridView
  - Other view collections





CREATE CONTEXT MENU

- When the registered view receives a long-click event, the system calls your onCreateContextMenu() method.

- ```
  @Override
  public void onCreateContextMenu(ContextMenu menu, View v,
                                  ContextMenuInfo menuInfo)
  {
      super.onCreateContextMenu(menu, v, menuInfo);
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.context_menu, menu);
  }
  ```

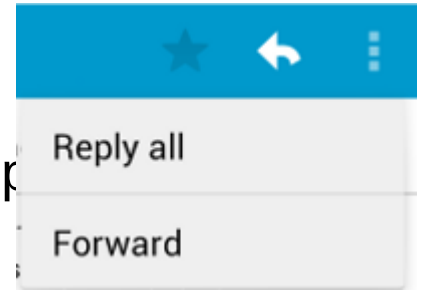
-
  ```

CREATE CONTEXT MENU

- When the user selects a menu item, the system calls this method onContextItemSelected()
-
```
@Override
public boolean onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info = (AdapterContextMenuInfo)
item.getMenuInfo();
    switch (item.getItemId()) {
        case R.id.edit:
            editNote(info.id);
            return true;
        case R.id.delete:
            deleteNote(info.id);
            return true;
        default:
            return super.onContextItemSelected(item);
    }
}
```

CREATE POPUP MENU

- A PopupMenu is a modal menu anchored to a View.
- It appears below the anchor view if there is room, or above the view otherwise. Touching outside of the popup will dismiss it.



  - Providing an overflow-style menu for actions that relate to specific content (such as Gmail's email headers).
  - Providing a second part of a command sentence (such as a button marked "Add" that produces a popup menu with different "Add" options).
  - Providing a drop-down similar to Spinner that does not retain a persistent selection.

CREATE POPUP MENU

- Popup Menu normally is created programmatically.
-
```
<ImageButton
    android:src="@drawable/ic_overflow_holo_dark"
    android:contentDescription="@string/descr_button"
    android:onClick="showPopup" />
```
- Popup menu is coded in Activity:
-
```
public void showPopup(View v) {
    PopupMenu popup = new PopupMenu(this, v);
    MenuInflater inflater = popup.getMenuInflater();
    inflater.inflate(R.menu.actions, popup.getMenu());
    popup.show();
}
```
-

CREATE POPUP MENU

- To perform an action when the user selects a menu item, you must
- Implement the `PopupMenu.OnMenuItemClickListener` interface
- Register it with your PopupMenu by calling `setOnMenuItemclickListener()`

```
• PopupMenu popup = new PopupMenu(this, v);
…..
• popup.setOnMenuItemClickListener(menuItemClickListener);
```

CREATE POPUP MENU

```
public class MenuItemClickListener implements
OnMenuItemClickListener {

 @Override
 public boolean onMenuItemClick(MenuItem item) {
     switch (item.getItemId()) {
         case R.id.archive:
             archive(item);
             return true;
         case R.id.delete:
             delete(item);
             return true;
         default:
             return false;
     }
 }
}
```

ANDROID INTENT

Intent in Android

ANDROID INTENT

- Intents are asynchronous messages which allow Android components to request functionality from other components of the Android system.
- Intents can be used to signal to the Android system that a certain event has occurred.
- Other components in Android can register to this event via an intent filter.

ANDROID INTENT

- Intents are send to the Android system via a method call
  - via the startActivity() method you can start activities.

- Depending on how the Intent was constructed the Android system will run an receiver determination and determine possible components which can be started.

- If several components have registered for the same intents the user can decide which component should be started

ANDROID EXPLICIT INTENT

- *Explicit intents* explicitly defines the component which should be called by the Android system, by using the Java class as identifier.

- *Explicit intents* are typically used within on application as the classes in an application are controlled by the application developer

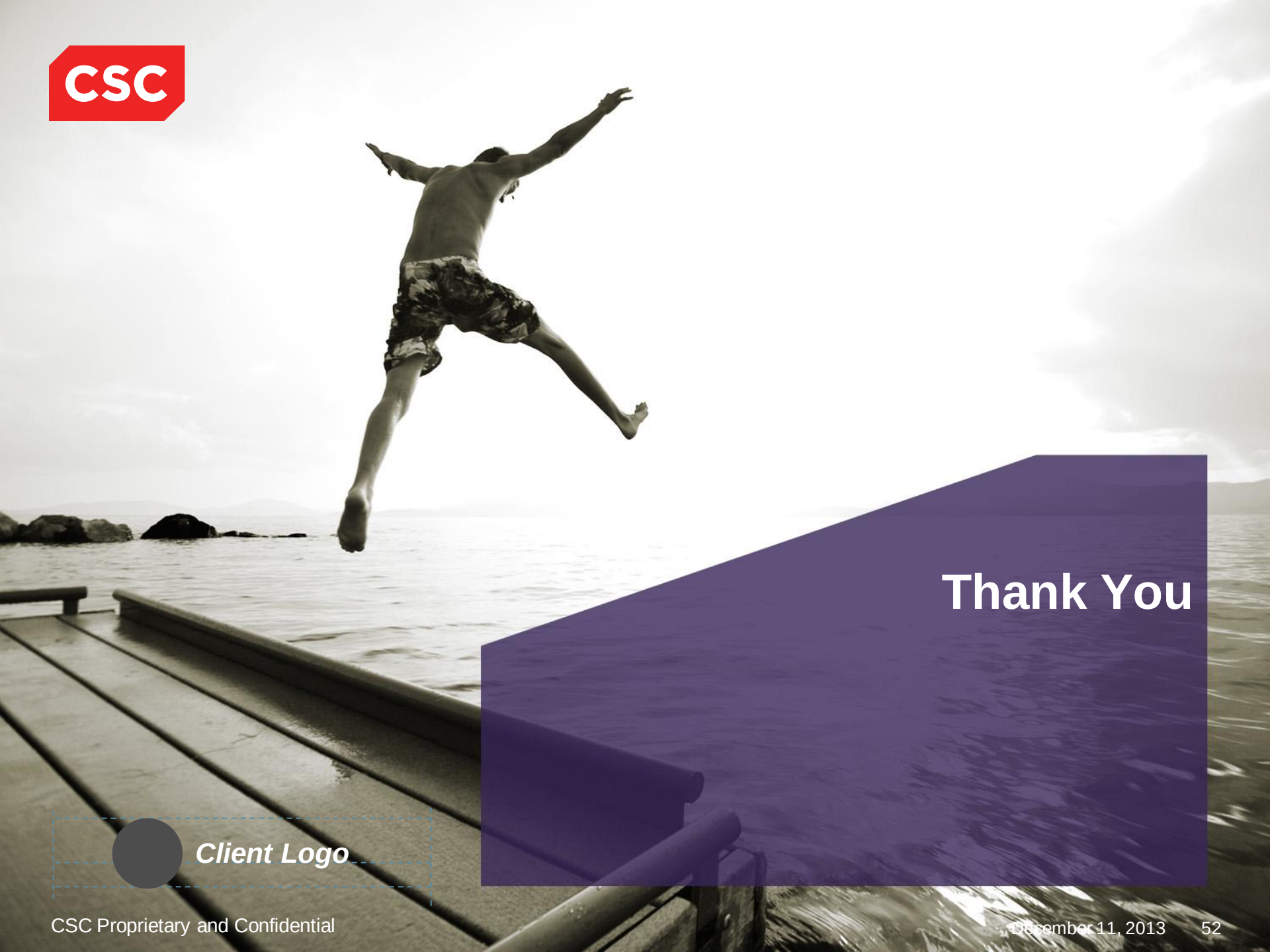
```
Intent i = new Intent(this, ActivityTwo.class);
i.putExtra("Value1", "This value one for ActivityTwo ");
i.putExtra("Value2", "This value two ActivityTwo");
startActivity(i);
```

ANDROID IMPLICIT INTENT

- *Implicit intents* specify the action which should be performed and optionally data which provides data for the action.
- *Intents* are send to the Android system it searches for all components which are registered for the specific action and the data type.
- If only one component is found, Android starts this component directly. If several components are identifier by the Android system, the user will get an selection dialog and can decide which component should be used for the Intent

```
Intent i = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.vogella.com"));
startActivity(i);
```

Q&A



Thank You

Client Logo

Revision History

Date	Version	Description	Updated by	Reviewed and Approved By
12 May 2013	0.5	Release for review	Nam Tu	


CSC
BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING