ANDROID PROGRAMMING WITH DATABASE

Nam Nguyen Tu
Linh Ha Nguyen

8-4-2013

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 the way to save data on Android
- Understand how to make an Android database application

Agenda

I.	Section One	xx
II.	Section Two	xx
III.	Section Three	xx
IV.	Section Four	xx
V.	Section Five	xx
VI.	Section Six	xx
VII.	Section Seven	xx

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 programming basic

Assessment Disciplines

- Class Participation: 70%
- Assignment: 30%
- Final Exam: 0%
- Passing Scores: 80%

Duration and Course Timetable

- Course Duration: <hrs>
- Course Timetable:
  - From <time> to <time>
  - Break <x> minutes from <time> to <time>

Further References

- <Source 1>
- <Source 2>
- …

Set Up Environment

- To complete the course, your PC must install:
  - Software 1
  - Software 2
  - …

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 STORAGE

Introduction about Android storage

ANDROID STORAGE

- Why do Android applications need data storage ?
  - Store application code
  - Store preferences
  - Store user-specific data

ANDROID STORAGE

- Why do Android applications need storage ?

	APPLICATION CODE	PERSISTENT DATA
	APP	DATA
	APP	DATA

ANDROID STORAGE OPTIONS

- Android provides several options for you to save persistent application data. The solution is chosen on needs:
  - Data is private to application ?
  - Accessible to other applications (and the user) ?
  - How much space your data requires ?

CONTENT PROVIDERS

- Android provides a way for you to expose even your private data to other applications — with a content provider:
  - An optional component that exposes read/write access to application data
  - Subject to whatever restrictions developer want to impose

ANDROID STORAGE OPTIONS

- Data storage options are :

- Shared Preferences
  - Store private primitive data in key-value pairs.
- Internal Storage
  - Store private data on the device memory.
- External Storage
  - Store public data on the shared external storage.
- SQLite Databases
  - Store structured data in a private database.
- Network Connection
  - Store data on the web with your own network server.

SHARED PREFERENCES

Save data to shared preferences

SHARED PREFERENCES

- Interface for accessing and modifying preference data
- provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types.
- You can use SharedPreferences to save any primitive data: booleans, floats, ints, longs, and strings.
- This data will persist across user sessions (even if your application is killed).provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use SharedPreferences to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).

INTERNAL AND EXTERNAL STORAGE

Save data to shared preferences

INTERNAL STORAGE

- You can save files directly on the device's internal storage. By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user). When the user uninstalls your application, these files are removed.

- To create and write a private file to the internal storage:
  - Call openFileOutput() with the name of the file and the operating mode. This returns a FileOutputStream.
  - Write to the file with write().
  - Close the stream with close().

INTERNAL STORAGE

- Saving cache files
  - If you'd like to cache some data, rather than store it persistently, you should use getCacheDir() to open a File that represents the internal directory where your application should save temporary cache files.
  - When the device is low on internal storage space, Android may delete these cache files to recover space. However, you should not rely on the system to clean up these files for you.

EXTERNAL STORAGE

- Every Android-compatible device supports a shared "external storage" that you can use to save files.
- This can be a removable storage media (such as an SD card) or an internal (non-removable) storage.
- Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.
- Saving files that should be shared
  - If you want to save files that are not specific to your application and that should not be deleted when your application is uninstalled, save them to one of the public directories on the external storage. These directories lay at the root of the external storage, such as Music/, Pictures/, Ringtones/, and others.

EXTERNAL STORAGE

- Saving cache files
  - If you're using API Level 8 or greater, use getExternalCacheDir() to open a File that represents the external storage directory where you should save cache files.
    - If the user uninstalls your application, these files will be automatically deleted. However, during the life of your application, you should manage these cache files and remove those that aren't needed in order to preserve file space.
  - If you're using API Level 7 or lower, use getExternalStorageDirectory() to open a File that represents the root of the external storage, then write your cache data in the following directory:
    - /Android/data/<package_name>/cache/

SQLITE DATABASE

SQLITE SUPPORT

- Android provides full support for SQLite databases.
- Any databases you create will be accessible by name to any class in the application, but not outside the application.

CREATE DATABASE

- The recommended method to create a new SQLite database is to create a subclass of SQLiteOpenHelper and override the onCreate() method, in which you can execute a SQLite command to create tables in the database.

READ AND WRITE DATABASE

- To write to and read from the database, call getWritableDatabase() and getReadableDatabase(), respectively. These both return a SQLiteDatabase object that represents the database and provides methods for SQLite operations.
- Execute SQLite queries using the SQLiteDatabase query() methods, which accept various query parameters, such as the table to query, the projection, selection, columns, grouping, and others.
- For complex queries, such as those that require column aliases, you should use SQLiteQueryBuilder,For complex queries, such as those that require column aliases, you should use SQLiteQueryBuilder,

READ DATABASE WITH CURSOR

- Every SQLite query will return a Cursor that points to all the rows found by the query. The Cursor is always the mechanism with which you can navigate results from a database query and read rows and columns.
- Every SQLite query will return a Cursor that points to all the rows found by the query. The Cursor is always the mechanism with which you can navigate results from a database query and read rows and columns.

OBJECT-RELATIONAL MAPPING

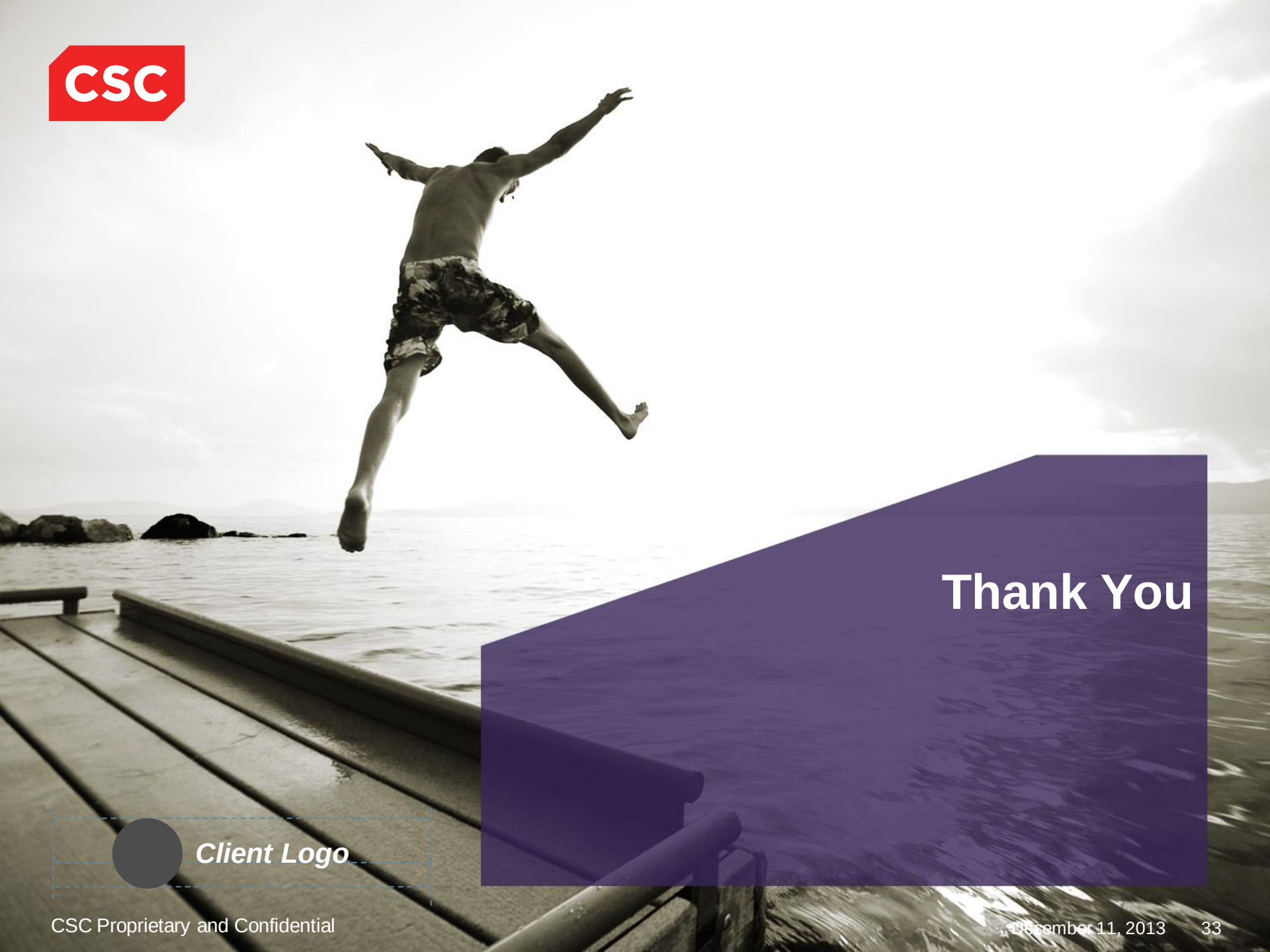
- There are some O/R mapping libraries that will help in development:
  - Ormlite: http://ormlite.com
  - greenDAO: http://greendao-orm.com
  - …

DATABASE DEBUGGING

- The Android SDK includes a sqlite3 database tool that allows you to browse table contents, run SQL commands, and perform other useful functions on SQLite databases.

Points to Remember

Q&A



Thank You

Client Logo

Revision History

Date	Version	Description	Updated by	Reviewed and Approved By


CSC
BUSINESS SOLUTIONS
TECHNOLOGY
OUTSOURCING