PERSISTENT DATA

Persistent Data

- Storage of data that is persistent across application invocations
- User preferences
- Application state
- Application data
- Databases

Android Data Storage

- Shared preferences
  - Private, primitive data in key-value pairs
  - User preferences
- Internal storage
  - Private data on device memory
- External storage
  - Public, SD card storage
- SQLite databases
- Network storage

iOS Data Storage

- Property lists
  - Collection of standard classes
- User defaults
  - Property list used for preferences
- File system storage
  - Sandbox
- SQLite databases
- Core Data

Android Preferences

- SharedPreferences class provides framework to save/retrieve persistent key-value pairs
  - Primitive data types
    - boolean, float, int, long, string
- PreferenceActivity extends ListActivity
  - Hierarchy of Preference objects
  - Automatically saves to default SharedPreferences
  - Defined by XML similar to layout
  - Retrieved using getDefaultSharedPreferences

PreferenceActivity

- Preference object types
  - CheckBoxPreference
  - ListPreference
  - EditTextPreference
  - RingtonePreference
  - PreferenceScreen
  - Preference
  - PreferenceCategory
- Attributes
  - android:key
  - android:title
  - android:defaultValue

Android File Storage

- Internal storage
  - Write

```
FileOutputStream openFileOutput (String
  name, int mode)
```

  - Read

```
FileInputStream openFileInput (String name)
```

  - Other useful methods
    - getFilesDir(), getDir(), deleteFile(), fileList()
  - Read, write, close using java.io methods
    - .write(), .read(), .close()
  - Alternatively, include file in res/raw/

```
InputStream openRawResource (int id)
```

Android File Storage

- External storage
  - Methods and constants provided through Environment class
  - Check availability

```
String getExternalStorageState()
```

  - Get directory for external storage using

```
File getExternalFilesDir (String type)
```

    - /<external_storage>/Android/data/<package>/files/
  - For public files (not deleted on uninstall)

```
File
  getExternalStoragePublicDirectory(Str
  ing type)
```

Android File Storage

- Example external storage sequence

```
File file = new File (getExternalFilesDir
   (null), “Demo.txt”);
OutputStream os = new FileOutputStream (file);
os.write (data);
```

iOS Data Storage

iOS Property Lists

- Collection of collections
  - NSArray, NSDictionary, NSNumber, NSString, NSDate, NSData
- Not really a class, but the SDK provides methods which operate only on Property Lists

  [plist writeToFile:(NSString *)path atomically:(BOOL)]
- Good for small amounts of data
  - Preferences
  - Settings

iOS Property Lists

- May be stored permanently
  - NSUserDefaults
  - XML, binary, ASCII (deprecated)
- NSPropertyListSerialization
  - NSData -> plist (propertyListWithData:options:format:error:)
  - plist -> NSData (dataFromPropertyList:format:options:error:)
- Write to file

  + (BOOL)writeToURL:(NSURL *)fileURL atomically:(BOOL)atomically
- Read from file

  + initWithContentsOfURL:(NSURL *)aURL

NSUserDefaults

- Lightweight storage of property lists
- Provides shared instance **standardUserDefaults**

  [NSUserDefaults standardUserDefaults]
- Example methods
  - Write

    - (void) setDouble:(double)aDouble forKey:(NSString *)key

    - (void) setObject:(id)obj forKey:(NSString *)key
  - Read

    - (NSInteger) integerForKey:(NSString *)key

    - (NSArray *) arrayForKey:(NSString *)key
- Need to synchronize to save batch of writes

```
[[NSUserDefaults standardUserDefaults]
  synchronize];
```

iOS File Storage

- Can only write within “sandbox”
  - Application bundle directory
    - NOT writeable
    - Binary, xibs, jpgs, etc.
  - Documents directory
    - Good location for permanent data
  - Cache directory
    - Temp files (not backed by iTunes)
  - Many more (see NSSearchPathDirectory)

iOS File Storage

- How to obtain path to sandbox directory

```
NSArray *NSSearchPathForDirectoriesInDomains (
        NSSearchPathDirectory directory,
        NSSearchPathDomainMask domainMask,
        BOOL expandTilde);
```

  - NSArray typically returns single item in iOS
    - Use lastObject method of NSArray

iOS File Storage

- NSFileManager
  - Provides utility operations for file system
    - Verify if file exists
    - Create directories
    - Get contents of directory
    - Move, copy, delete files
- NSString
  - Provides useful path string utilities and read/write to file
    - (NSString *) stringByAppendingPathComponent:
    - (NSString *) stringByDeletingLastPathComponent
    - (BOOL) writeToFile:atomically:encoding:error:
    - (NSString *) stringWithContentsOfFile:usedEncoding:error:

SQLite

- SQL in a single file
  - Fast, low memory, reliable, open source
- C/C++ API

```
int sqlite3_open(const char *filename, sqlite3 **db)
int sqlite3_exec(sqlite3 *db, const char *sql, int (*callback), void
    *content, char **error)
int mycallback(void *context, int count, char **values, char **cols)
int sqlite3_close(sqlite3 *db)
```

To be continued . . .

- Core Data