Options for storing application data permanently

- Shared preferences
  - Key-Value -pairs, use specific API (see http://developer.android.com/reference/android/content/SharedPreferences.html)
- Internal and external storage
  - Read/write files using normal Java file I/O into either device memory or external memory (memory card). (Note permission needed for the latter option)
- Network
  - http requests
- SQLite
- See http://developer.android.com/guide/topics/data/data-storage.html for overview of the options

SQLite

- Open source (at least almost) industry-standard SQL database for mobile and embedded environments
  - Lightweight
  - Robust
- Implemented as a library - no separate database management process(es)
  - Database is not shared between applications

SQLiteOpenHelper

- Inherit from this class to create an application-specific database helper
  - Database creation, initialization, upgrade, downgrade support

```
public class CitiesOpenHelper extends SQLiteOpenHelper {

    static final String DATABASE_NAME = "cityDB";
    static final int DATABASE_VERSION = 5;
    static final String CITIES_TABLE_NAME = "cities";
    static final String KEY_CITYNAME = "name";
    static final String KEY_CITYWWW = "www";
    static final String KEY_CITYSTARS = "stars";

    private static final String CITIES_TABLE_CREATE =
                "create table " + CITIES_TABLE_NAME + " (" +
                "_id integer primary key autoincrement, " +
                KEY_CITYNAME + " text not null," +
                KEY_CITYWWW + " text, " +
                KEY_CITYSTARS + " integer" +
                ");";

    public CitiesOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        Log.d("CitiesOpenHelper", "CitiesOpenHelper()");
    }
```

used by onUpgrade()

_id autoincremented primary key recommended, obligatory if you implement content provider for the data

Example - onCreate()

```
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CITIES_TABLE_CREATE);

    Log.d("CitiesOpenHelper", "onCreate");

    ContentValues values = new ContentValues();

    values.put(KEY_CITYNAME, "Hamina");
    values.put(KEY_CITYWWW, "www.hamina.fi");
    values.put(KEY_CITYSTARS, 1);

    db.insert(CITIES_TABLE_NAME, null, values);
    values.clear();

    values.put(KEY_CITYNAME, "Hong Kong");
    values.put(KEY_CITYWWW, "www.gov.hk/en");
    values.put(KEY_CITYSTARS, 5);
    db.insert(CITIES_TABLE_NAME, null, values);
    values.clear();
   ...
}
```

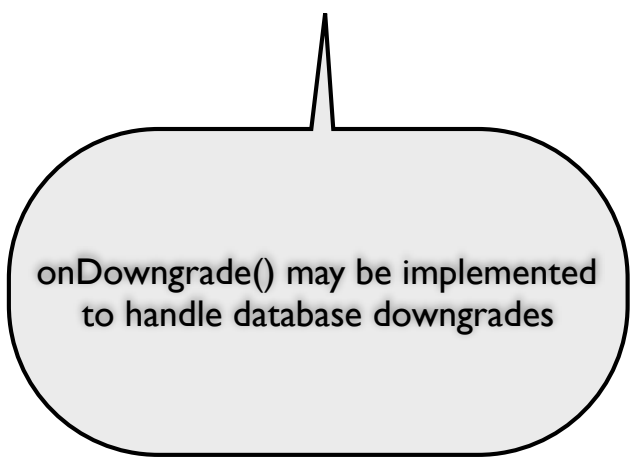
Create database

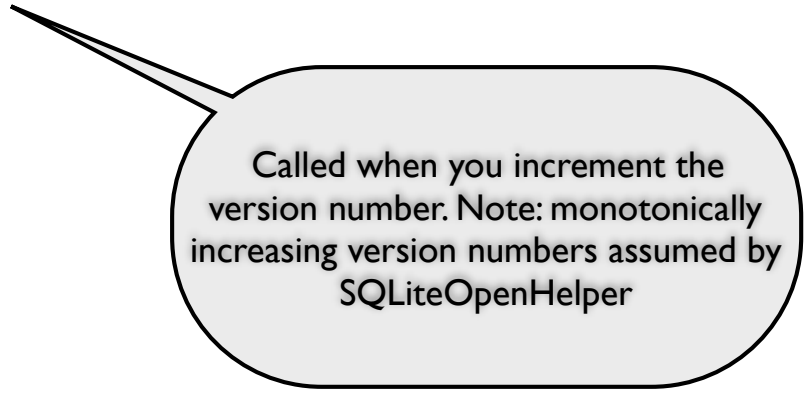
Initialization combined with creation

Insert ContentValues objects to database

Example - onUpgrade()

```
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    Log.d("CityView", "Upgrading database from version " + oldVersion + " to "
           + newVersion + ", which will destroy all old data");
    db.execSQL("DROP TABLE IF EXISTS " + CITIES_TABLE_NAME);
    onCreate(db);
}
```





- Note that normally you would do something more complicated (and application data -specific) in case of upgrade!

Content Providers

- Way to provide access to data across separate applications
  - remember - applications are separate, no shared memory etc.
  - another way is IPC mechanism with services
- Provider takes care of storing its data in a way it likes
  - data is exposed as a simple table
  - ... and often DB is used for implementing data storage
- With some convenience classes (CursorLoader) using content provider inside a single application can be easiest option, too

Creating a Content Provider

- Subclass `ContentProvider` and implement
  - `query()` - mandatory
    `insert()`
    `update()`
    `delete()`
    `getType()` - mandatory
    `onCreate()` - mandatory
- interface exposed is SQL like
- Declare your provider in manifest!

Example - provide cities database in cityview app

```
public class MyProvider extends ContentProvider {
  SQLiteDatabase thisDB;

  public static final String PROVIDER_NAME = "com.example.cities";
  public static final Uri CONTENT_URI = Uri.parse("content://" + PROVIDER_NAME + "/cities");

  public static final String _ID = "_id";
  public static final String CITY = "name";

  private static final int CITIES = 1;
  private static final int CITY_ID = 2;

  private static final UriMatcher uriMatcher;
    static {
       uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
       uriMatcher.addURI(PROVIDER_NAME, "cities", CITIES);
       uriMatcher.addURI(PROVIDER_NAME, "cities/#", CITY_ID);
    }

  public int delete(Uri uri, String selection, String[] selectionArgs) {
    return 0;
  }

  public String getType(Uri uri) {
    switch (uriMatcher.match(uri)) {
      case CITIES:
        return "vnd.android.cursor.dir/vnd.example.cities ";
      case CITY_ID:
        return "vnd.android.cursor.item/vnd.example.cities ";
      default:
        throw new IllegalArgumentException("Unsupported URI: " + uri);
    }
  }
```

Provider Example cont.

```
public Uri insert(Uri uri, ContentValues values) {
  return null;
}

public boolean onCreate() {
  Context c = getContext();
  CitiesOpenHelper dbHelper = new CitiesOpenHelper(c);
  thisDB = dbHelper.getReadableDatabase();
  if (thisDB == null)
    return false;
  else
    return true;
}

public Cursor query(Uri uri, String[] projection, String selection,
                    String[] selectionArgs, String sortOrder) {
  SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();
  sqlBuilder.setTables(CitiesOpenHelper.CITIES_TABLE_NAME);

  if (uriMatcher.match(uri) == CITY_ID)
    sqlBuilder.appendWhere(_ID + " = " + uri.getPathSegments().get(1));

  Cursor cur = sqlBuilder.query(thisDB, projection, selection, selectionArgs, null, null, sortOrder);

  cur.setNotificationUri(getContext().getContentResolver(), uri);
  return cur;
}

public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs) {
  return 0;
}
```

Reading list

- http://developer.android.com/training/basics/data-storage/databases.html
- http://developer.android.com/guide/topics/providers/content-providers.html
  - including Content Provider Basics, Creating a Content Provider, Calendar Provider, Contacts Provider