1. ORMLite Annotations

1.1 Define Database Table

With ORMLite annotations, for each of the Java classes that you would like to persist to your SQL database, you will need to add the @DatabaseTable annotation right above the public class line. Each class marked with one of these annotations will be persisted into its own database table. For example:

```
@DatabaseTable(tableName = "accounts")
public class Account {
…
```

See: http://ormlite.com/javadoc/ormlite-core/doc-files/ormlite_2.html#Class-Setup

Note: Entity class must have a public non-argument constructor. Otherwise, ORMLite and Simple XML are not able to create object automatically.

Note: Entity class of FLAX project should extends BaseEntity, especially exercise detail class should extends BaseExerciseDetail class, page item class should extends BasePage class.

1.2 Define Table Column

For each of the classes, you will need to add a @DatabaseField annotation to each of the fields in the class that are to be persisted to the database. Each field is persisted as a column of a database row. For example:

```
@DatabaseTable(tableName = "accounts")
public class Account {

    @DatabaseField(id = true)
    private String name;

    @DatabaseField(canBeNull = false)
    private String password;
    …
```

See: http://ormlite.com/javadoc/ormlite-core/doc-files/ormlite_2.html#Class-Setup

See: http://ormlite.com/javadoc/ormlite-core/doc-files/ormlite_2.html#Id-Column

Note: In FLAX project every table should have a unique id, consider uniqueness of id field when implement an entity. Otherwise will cause exception when saving data into database. Normally, if entity does not contains a field with unique value, we can use a generated id. (If entity does not contains a field with unique value, and cannot use generated id here, maybe should consider “Composite Id” see 1.5 Table with Composite Id)

1.3 Foreign Object (One-to-one relationship)

ORMLite supports the concept of "foreign" objects where one or more of the fields correspond to an object are persisted in another table in the same database. For example, if you had an Order objects in your database and each Order had a corresponding Account object, then the Order object would have foreign Account field. With foreign objects, just the id field from the Account is persisted to the Order table as the column "account_id". For example, the Order class might look something like:

```
@DatabaseTable(tableName = "orders")
```

```
public class Order {

        @DatabaseField(generatedId = true)
        private int id;

        @DatabaseField(foreign = true, foreignAutoRefresh = true)
        private Account account;
        …
```

See: http://ormlite.com/javadoc/ormlite-core/doc-files/ormlite_2.html#Foreign-Objects

Note: In FLAX project, the argument of foreign field’s @DatabaseField annotation should always set foreignAutoRefresh = true, in order to generate object field when querying data from database. Otherwise, when load an Order object data from database, the account field will be null.

1.4 Foreign Collection (One-to-many relationship)

A foreign collection allows you to add a collection of orders on the account table. Whenever an Account object is returned by a query or refreshed by the DAO, a separate query is made over the order table and a collection of orders is set on the account. All of the orders in the collection have a corresponding foreign object that matches the account. For example:

```
public class Account {
        …
        @ForeignCollectionField(eager = true, maxEagerLevel = MAX_EAGER_LEVEL)
        Collection<Order> orders;
        …

public class Order {
         @DatabaseField(generatedId=true)
         private int id;

         @DatabaseField(foreign = true, foreignAutoRefresh = true, columnName="account_id")
         private Account account;
        …
```

See: http://ormlite.com/javadoc/ormlite-core/doc-files/ormlite_2.html#Foreign-Collection

Note: In FLAX project, the argument of foreign field’s @ForeignCollectionField annotation should always set eager = true and maxEagerLevel = GlobalConstant.MAX_EAGER_LEVEL, in order to generate eager collection when querying data from database. This means when calling Get Method (e.g. getOrders()) to get a collection, will return the same collection without querying database. Otherwise (eager = false), when get Order collection data from an Account object multiple times, the Get Method will query database and return different collection objects, and that could be problematic when binding data with Android components.

Note: In FLAX project, all foreign collection should use java.uitl.Collection as data type, in order to make an integrate ORMLite and Simple XML

1.5 Table with Composite Id

Currently, ORMLite doesn't support Composite Id (Multiple primary keys), but we can simulate that, use an extra field which returns a combination of all primary keys as unique id. In order to do this, set useGetSet = true is necessary. Implement Get/Set methods, make a unique id with two or more fields. Making Set method as Deprecated, to warn user they are not supposed to use this method. It will only be called by ORMLite libaray.

```
    public class Order {
            @DatabaseField(id = true, useGetSet = true)
            private String id;

            public String getId() {
                    return name + "@" + url;
            }

            @Deprecated
            public void setId(String id) {
                    this.id = id;
            }
            …
```

See: http://ormlite.com/javadoc/ormlite-core/doc-files/ormlite_2.html#Local-Annotations

2. Simple XML Annotations

2.1 Parsing a Simple XML

User @Root annotate the class which represent root node of XML, @Element annotate the field which represent child node, @Attribute annotate the field which represent the attribute of the node. @Text annotate the field which represent the inner text of the node. For example:

```
    @Root(name="root")
    public class Example {

        @Element(name="message")
        private String text;

        @Attribute(name="id")
        private int index;
    …
```

corresponding to XML:

```
    <root id="123">
        <message>Example message</message>
    </root>
```

See: http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php#start

Parsing XML into nested object

See: http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php#nested

2.2 Parsing a List of Element

In FLAX, XML usually deal with a list of children node. In this case, should use @ElementList annotation, set inline = true and entry = children node's name.

```
@Root
public class PropertyList {

    @ElementList(inline = true, entry = “entry”)
    private Collection<Entry> list;

    @Element
    private String name;
}
@Root
public class Entry {

     @Attribute
     protected String key;

     @Element
     protected String value;
}
```

corresponding to XML:

```
<propertyList>
    <name>example</name>
    <entry key="one">
        <value>first value</value>
    </entry>
    <entry key="two">
        <value>second value</value>
    </entry>
    <entry key="three">
        <value>third value</value>
    </entry>
</propertyList>
```

See: http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php#inline
Different between inline list and normal list
See: http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php#list

2.3 Build Object Relation (Receiving Callbacks)

@Commit annotation can be used to receive callbacks from the parser. Declare a method annotated with @Commit. Then the method will be called after this object was done parsing. In FLAX, we can use it to build object relation (foreign relation). For example:

```
@DatabaseTable(tableName = "exerciselist_category")
public class Category {
    @ForeignCollectionField(eager = true, maxEagerLevel = MAX_EAGER_LEVEL)
    @ElementList(inline = true, entry = "exercise")
    private Collection<Exercise> exercises;

    @Commit
    private void buildRelation() {
        for (Exercise exercise : exercises) {
            exercise.setCategory(this);
        }
    }
…
```

Note: Recommend to use private method in order to avoid misuse.
See: http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php#callback

2.4 Reduce Nested Object Layer (XPath & JavaBean)

See: http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php#xpath
See: http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php#javabean