Siminov Framework Version 0.9-Beta Getting Started Guide

Siminov Framework
http://www.siminov.github.com/android-orm

Contents

Preface

Working with both Object-Oriented software and Relational Databases can be cumbersome and time consuming. Development costs are significantly higher due to a paradigm mismatch between how data is represented in objects versus relational databases. Siminov is an Object/Relational Mapping solution for Android environments. The term Object/Relational Mapping refers to the technique of mapping data from an object model representation to a relational data model representation (and visa versa). See `http://en.wikipedia.org/wiki/Object-relational_mapping` for a good high-level discussion.

> **Note**
> While having a strong background in SQL is not required to use Android-Siminov, having a basic understanding of the concepts can greatly help you understand Siminov more fully and quickly. Probably the single best background is an understanding of data modeling principles. You might want to consider these resources as a good starting point: `http://en.wikipedia.org/wiki/Data_modeling`

Siminov not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities. It can significantly reduce development time otherwise spent with manual data handling in SQLite. Siminov design goal is to relieve the developer from 99

Siminov may not be the best solution for data-centric applications that only use stored-procedures to implement the business logic in the database, it is most useful with object-oriented domain models and business logic in the Java-based. However, Siminov can certainly help you to remove or encapsulate vendor-specific SQLite code and will help with the common task of result set translation from a tabular representation to a graph of objects.

0.1 Get Involved

Use Siminov and report any bugs or issues you find. See `https://github.com/Siminov/android-orm/issues` for details. Try your hand at fixing some bugs or implementing enhancements. Again, see `https://github.com/Siminov/android-orm/issues`. Engage with the community using mailing lists, forums, IRC, or other ways listed at `https://github.com/Siminovl`. Help improve or translate this documentation. Contact us on the developer mailing list if you have interest.Spread the word. Let the rest of your organization know about the benefits of Siminov.

0.2 Getting Started Guide

New users may want to first look through the Siminov Getting Started Guide for basic information as well as tutorials. Even seasoned veterans may want to considering perusing the sections pertaining to build artifacts for any changes.

Chapter 1

Obtaining Siminov

1.0.1 Release Bundle Downloads

The Siminov Framework team provide release bundles hosted at **Github** File Release System, in ZIP and TGZ formats. Each release bundle contains JARs, documentation, source code, and other goodness.

You can download releases of Siminov Framework, in your chosen format, from the list at `http://siminov.github.io/android-orm/index.html`.

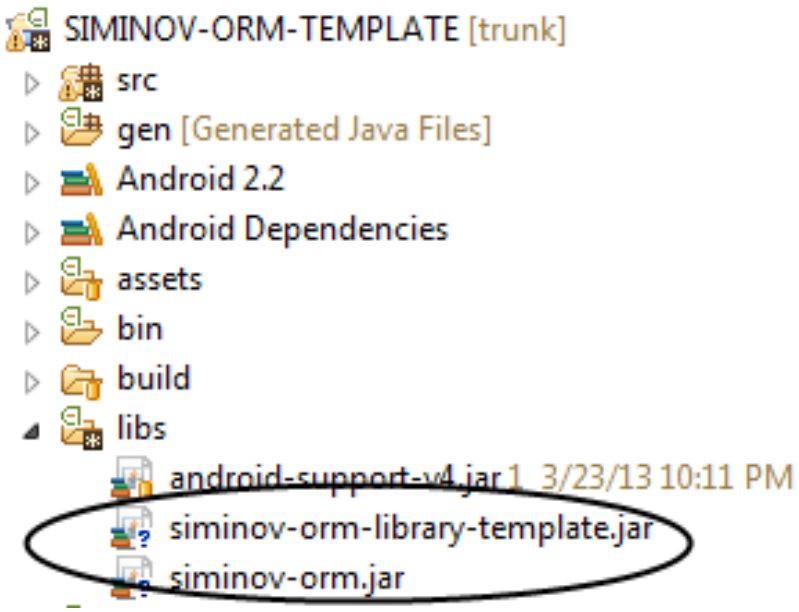
1. The lib/required/ directory contains all the JARs Siminov requires. All the jars in this directory must also be included in your project's classpath.

Chapter 2

Setting Up Application

1. Download Siminov jar from `http://siminov.github.io/android-orm/builds.html`
2. Add Siminov jar into your application libs folder.

   Example: Siminov Template Application



Chapter 3

Building Application

Building application based on Siminov Framework is easy, simple. By the help of **Siminov Template Application** we will explain how we can build application.

3.1 Configure ApplicationDescriptor.si.xml

Application Descriptor is the one who connects application to Siminov Framework. It provide basic information about application, which defines the behaviour of application.

```
<siminov>

  <property name="name">SIMINOV TEMPLATE</property>
  <property name="description">Siminov Template Application</property
      >
  <property name="version">0.9</property>

  <property name="load_initially">true</property>

  <!-- DATABASE-DESCRIPTORS -->
  <database-descriptors>
    <database-descriptor>DatabaseDescriptor.si.xml</database-
        descriptor>
  </database-descriptors>


  <!-- SIMINOV EVENTS -->
  <event-handlers>
      <event-handler>siminov.orm.template.events.SiminovEventHandler<
          /event-handler>
      <event-handler>siminov.orm.template.events.DatabaseEventHandler
          </event-handler>
```

```
  </event-handlers>

</siminov>
```

3.2 Configure DatabaseDescriptor.si.xml

Database Descriptor is the one who de

nes the schema of database.

```
<database-descriptor>

  <property name="database_name">SIMINOV-TEMPLATE</property>
  <property name="description">Siminov Template Database Config</
      property>
  <property name="is_locking_required">true</property>
  <property name="external_storage">false</property>
  <property name="database_implementer"></property>
  <property name="password"></property>

  <database-mappings>
    <database-mapping path="Liquor-Mappings/Liquor.si.xml" />
    <database-mapping path="Liquor-Mappings/LiquorBrand.si.xml" />
  </database-mappings>

  <libraries>
    <library>siminov.library.template.resources</library>
  </libraries>

</database-descriptor>
```

3.3 Configure LibraryDescriptor.si.xml

Library Descriptor is the one who de

nes the properties of library.

```
<library>

  <property name="name">SIMINOV LIBRARY TEMPLATE</property>
```

```
  <property name="description">Siminov Library Template</property>

  <database-mappings>
    <database-mapping path="Credential.si.xml" />
  </database-mappings>

</library>
```

3.4 Configure DatabaseMappingDescriptor.si.xml

Database Mapping Descriptor is one which does ORM, it maps POJO class to database table.

```
<database-mapping>

  <table table_name="LIQUOR" class_name="siminov.orm.template.model.
      Liquor">

    <column variable_name="liquorType" column_name="LIQUOR_TYPE">
      <property name="type">java.lang.String</property>
      <property name="primary_key">true</property>
      <property name="not_null">true</property>
      <property name="unique">true</property>
    </column>

    <column variable_name="description" column_name="DESCRIPTION">
      <property name="type">java.lang.String</property>
    </column>

    <column variable_name="history" column_name="HISTORY">
      <property name="type">java.lang.String</property>
    </column>

    <column variable_name="link" column_name="LINK">
      <property name="type">java.lang.String</property>
      <property name="default">www.wikipedia.org</property>
    </column>

    <column variable_name="alcholContent" column_name="ALCHOL_CONTENT
        ">
      <property name="type">java.lang.String</property>
    </column>

    <index name="LIQUOR_INDEX_BASED_ON_LINK" unique="true">
```

```
      <column>HISTORY</column>
    </index>

    <relationships>

        <one-to-many refer="liquorBrands" refer_to="siminov.orm.
            template.model.LiquorBrand" on_update="cascade" on_delete=
            "cascade">
        <property name="load">true</property>
      </one-to-many>

    </relationships>

  </table>

</database-mapping>
```

3.5 Using Annotation

Liquor pojo class of Siminov template application.

```
@Table(tableName=Liquor.TABLE_NAME)
@Indexes({
  @Index(name="LIQUOR_INDEX_BASED_ON_LINK", unique=true, value={
    @IndexColumn(column=Liquor.LINK)
  }),
})
public class Liquor extends Database implements Serializable {

  //Table Name
  transient public static final String TABLE_NAME = "LIQUOR";

  //Column Names
  transient public static final String LIQUOR_TYPE = "LIQUOR_TYPE";
  transient public static final String DESCRIPTION = "DESCRIPTION";
  transient public static final String HISTORY = "HISTORY";
  transient public static final String LINK = "LINK";
  transient public static final String ALCHOL_CONTENT = "
     ALCHOL_CONTENT";

  //Liquor Types
  transient public static final String LIQUOR_TYPE_GIN = "Gin";
  transient public static final String LIQUOR_TYPE_RUM = "Rum";
  transient public static final String LIQUOR_TYPE_TEQUILA = "Tequila
     ";
```

```
transient public static final String LIQUOR_TYPE_VODKA = "Vodka";
transient public static final String LIQUOR_TYPE_WHISKEY = "Whiskey
    ";
transient public static final String LIQUOR_TYPE_BEER = "Beer";
transient public static final String LIQUOR_TYPE_WINE = "Wine";


//Variables

@Column(columnName=LIQUOR_TYPE,
     properties={
       @ColumnProperty(name=ColumnProperty.PRIMARY_KEY, value="true"
           ),
       @ColumnProperty(name=ColumnProperty.NOT_NULL, value="true"),
       @ColumnProperty(name=ColumnProperty.UNIQUE, value="true")
       })
private String liquorType = null;

@Column(columnName=DESCRIPTION)
private String description = null;

@Column(columnName=HISTORY)
private String history = null;

@Column(columnName=LINK,
     properties={
       @ColumnProperty(name=ColumnProperty.DEFAULT, value="www.
           wikipedia.org")
       })
private String link = null;

@Column(columnName=ALCHOL_CONTENT)
private String alcholContent = null;

@OneToMany(onUpdate="cascade", onDelete="cascade",
     properties={
       @RelationshipProperty(name=RelationshipProperty.LOAD, value="
           true")
  })
private ArrayList<LiquorBrand> liquorBrands = null;

//Methods

public String getLiquorType() {
  return this.liquorType;
}

public void setLiquorType(String liquorType) {
  this.liquorType = liquorType;
```

```
  }

  public String getDescription() {
    return this.description;
  }

  public void setDescription(String description) {
    this.description = description;
  }

  public String getHistory() {
    return this.history;
  }

  public void setHistory(String history) {
    this.history = history;
  }

  public String getLink() {
    return this.link;
  }

  public void setLink(String link) {
    this.link = link;
  }

  public String getAlcholContent() {
    return this.alcholContent;
  }

  public void setAlcholContent(String alcholContent) {
    this.alcholContent = alcholContent;
  }

  public ArrayList<LiquorBrand> getLiquorBrands() {
    return this.liquorBrands;
  }

  public void setLiquorBrands(ArrayList<LiquorBrand> liquorBrands) {
    this.liquorBrands = liquorBrands;
  }
}
```

3.6 Saving Entity Object

Saving liquor entity object.

```
Liquor beer = new Liquor();
beer.setLiquorType("Beer");
beer.setDescription("Beer Description");
beer.setHistory("Beer History");
beer.setLink("Beer Web Link");
beer.setAlcholContent("Beer Alchol Content");

try {
  beer.save();
} catch(DatabaseException databaseException) {
  //Log It.
}
```

3.7 Updaing Entity Object

Updating liquor object.

```
Liquor beer = new Liquor();
beer.setLiquorType("Beer");
beer.setDescription("Beer Description");
beer.setHistory("Beer History");
beer.setLink("Beer Web Link");
beer.setAlcholContent("Beer Alchol Content");

try {
  beer.update();
} catch(DatabaseException databaseException) {
  //Log It.
}
```

3.8 Save Or Update Entity Object

Save or update liquor object.

```
Liquor beer = new Liquor();
beer.setLiquorType("Beer");
```

```
    beer.setDescription("Beer Description");
    beer.setHistory("Beer History");
    beer.setLink("Beer Web Link");
    beer.setAlcholContent("Beer Alchol Content");

    try {
      beer.saveOrUpdate();
    } catch(DatabaseException databaseException) {
      //Log It.
    }
```

3.9 Deleting Entity Object

Deleting liquor object.

```
    Liquor beer = new Liquor();
    beer.setLiquorType("Beer");
    beer.setDescription("Beer Description");
    beer.setHistory("Beer History");
    beer.setLink("Beer Web Link");
    beer.setAlcholContent("Beer Alchol Content");

    try {
      beer.delete();
    } catch(DatabaseException databaseException) {
      //Log It.
    }
```