qabel-core Documentation

Release 0.26.2

Nermin Nicevic

November 07, 2016

Contents

The Qabel Core is parted in 3 main Modules:

CHAPTER 1

Core Module

1.1 Accounting

The accounting packages contains Information about a users accounting profile and storage quota.

The BoxClient is the collaboration of Account related use-cases. It needs an HTTP client to open the connection to the AccountingServer.

Following Use-Cases are provided by the accounting package:

BoxClient:

- Register(create) an Account
- Update account information (such as resetting password)
- Login / Authentication
- List of Identitys
- **retrieve and create various information related to the Box Storage**
  - QuotaState, storage information
  - prefixes for connecting into the Box Volumes

1.1.1 Usage Requirements

- valid instance of AccountingServer
- valid instance of AccountingProfile
- *only for tests: instance of CloseableHttpClient*

1.2 Config

The Config folder/package contains many configuration classes and some few general entity's.

Here u sould find things like:

Entities:

- Account related Entity stuff, and even TypeAdapter for import or exporting those
- Identity,

- AccountingServer, DropServer definitions
- Import / Export related stuff for sync settings
- The Observer implementation for some entities such as for the Identity

You shouldn't find just raw entity classes but also some factory which are probably the most used in here.

1.3 Contacts

The Contacts module represents an contact related to one or more identity.

ContactExchangeFormats can be found in here as well.

- import / export of one contact into given format
- import / export of a list of contacts

1.4 Crypto

Welcome to the crypto section. Here are our primitive crypto wrapper classes and implementations of the current used crypto curve.

Everything in here is most likely nothing you should touch so easily and they won't change anytime soon. There is no need to use stuff in here directly.

For example: the QblECKeyPair stuff is in here they are used in identity's and contacts.

1.5 Drop

The Drop Package contains various stuff for the communication part of Qabel. Entitiy's and parser for DropMessages are in here.

Those Use-Cases u should find in the drop package:

- Communication related methods between Application and DropServer
- Import / export of DropMessages
- Creating and retrieving of DropMessages (encrypting and decrypting algorithms)
- MetaData of DropMessages

Some of classes of the crypto package are used in here.

1.6 Event

The Event package contains the core EventDispatcher for some own events like Identities changed or Identity changed observables.

Usually a Specific client like the Android Client has its own specific dispatchers.

1.7 Extensions

The Extensions package is some kind of Utils package but contains only Kotlin extension methods to extend some some util functionality without the need of changing or modifying some Interfaces to apply some class methods.

such as for example:

```
fun <T  : Entity> Set<T>.findById(id : Int) = find { it.id == id }
```

So every Instance of an 'Entity' can use this findById method.

1.8 HTTP

This package intend is to control and configure various http actions.

Currently it just holds an preset of an generalized HTTPResult which is used in Drop communication.

The HTTP package may grow soon.

1.9 Index

Index package represents the implementation of the Index Service.

Containing:

- API / Server connection to the index server
- CRUD Index
- The transmission of the given data provided by the CURD is encrypted via noise box

1.10 Logging

The logging package is self explaining.

Here is the Inteface of the logger for an short overview:

```
fun trace(msg: Any, vararg args: Any)
fun debug(msg: Any, vararg args: Any)
fun info(msg: Any, vararg args: Any)
fun warn(msg: Any, vararg args: Any)
fun error(msg: Any?, exception: Throwable?)
```

1.11 Repository

The Repository package is one of our biggest packages.

All database communication is stored here. It has an implementation with a sqlite database.

1.11.1 Subpackages

Entities

Currently only the Dropstate. Probably be modified soon.

Exception

Various repository related exceptions are located in here. Some examples:

Used when persisting failed

```
public class PersistenceException extends Exception {
    public PersistenceException(String message) {
        super(message);
    }

    public PersistenceException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Entity duplication found? throw this:

```
class EntityExistsException(val msg: String = "Entity already exists") : PersistenceException(msg) {

}
```

If u search for a given entity and its not there you can thorw this:

```
public class EntityNotFoundException extends Exception {
    public EntityNotFoundException(String message) {
        super(message);
    }

    public EntityNotFoundException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Transaction failures? throw this:

```
public class TransactionException extends PersistenceException {
    public TransactionException(String message) {
        super(message);
    }

    public TransactionException(String message, Throwable cause) {
        super(message, cause);
    }
}
```

Framework

Here you find our small database framework. Usually no changes in here. If u intend to modify some data schema then go to the *SQLite* package.

SQLite

This is the the biggest subpackage. Here u find the implementation of the database abstraction layer, each schemas, migrations and hydrator. The explicit SQLite repositorys are located in here.

Some examples of Repositorys: SqliteAccountRepository - which is used to store account and authentication information SqllieContactRepository - which obviously is the repository for contacts

---

Note: If u change something on the database structure - for example adding an new column than u need to create a migration. Samples can be found in sqlite/migration package.

1.12 Ui

The Ui packages is currently used for buffered view loading in android but is abstract enough to be in core ;)

Note: EntityUIExtensions.kt is a ui related utils class for retrieving initials of an alias or make the keyIdentifier human readable and pretty.

1.13 Util

Well this is kinda self explaning right? This is the place for some rally general util stuff.

One example:

Lazy loading HashMap

```
package de.qabel.core.util;

import java.util.HashMap;

public class LazyHashMap<K, V> extends HashMap<K, V> implements LazyMap<K, V> {
    @Override
    public V getOrDefault(K key, CheckedFunction<K, V> defaultValueFactory) {
        synchronized (this) {
            if (!containsKey(key)) {
                try {
                    put(key, defaultValueFactory.apply(key));
                } catch (Exception e) {
                    throw new RuntimeException(e.getMessage(), e);
                }
            }
            return get(key);
        }
    }
}
```

CHAPTER 2

Box Module

2.1 Http

2.1.1 Cache

2.1.2 Command

2.1.3 Dto

2.1.4 Exception

2.1.5 Factory

2.1.6 Hash

2.1.7 Jdbc

2.2 Http

CHAPTER 3

Chat Module

3.1 Repository Package

This package contains the definitions and implementation of the chat's database layer and repository methods. Like all our database schema it has its own migration scripts.

3.2 Service Package

The service represents the whole chat and chat sharing. It uses the Box(for sharing) and Drop(for messages):

https://qabel.github.io/docs/Qabel-Protocol-Box/

3.2.1 Basic Features

- **Send and receive messages**
  - **Prerequisites: (each object has its own local repository)**
    - Identity and Contact
    - ChatDropMessage (type = MESSAGE)
    - ChatService to send and receive
- **Send and receive shares to an contact**
  - **Prerequisites:**
    - Identity and Contact
    - BoxFile
    - BoxVolume to upload and navigate the BoxFile
    - SharingService to send and receive