3.2 Building Queries

There are a couple of different ways that you can build queries. The `QueryBuilder` has been written for ease of use as well for power users. Simple queries can be done linearly:

```
QueryBuilder<Account, String> queryBuilder =
  accountDao.queryBuilder();
// get the WHERE object to build our query
Where<Account, String> where = queryBuilder.where();
// the name field must be equal to "foo"
where.eq(Account.NAME_FIELD_NAME, "foo");
// and
where.and();
// the password field must be equal to "_secret"
where.eq(Account.PASSWORD_FIELD_NAME, "_secret");
PreparedQuery<Account> preparedQuery = queryBuilder.prepare();
```

The SQL query that will be generated from the above example will be approximately:

```
SELECT * FROM account
  WHERE (name = 'foo' AND password = '_secret')
```

If you'd rather chain the methods onto one line (like StringBuilder), this can also be written as:

```
queryBuilder.where()
  .eq(Account.NAME_FIELD_NAME, "foo")
  .and()
  .eq(Account.PASSWORD_FIELD_NAME, "_secret");
```

If you'd rather use parenthesis to group the comparisons properly then you can call:

```
Where<Account, String> where = queryBuilder.where();
where.and(where.eq(Account.NAME_FIELD_NAME, "foo"),
          where.eq(Account.PASSWORD_FIELD_NAME, "_secret"));
```

All three of the above call formats produce the same SQL. For complex queries that mix ANDs and ORs, the last format may be necessary to get the grouping correct. For example, here's a complex query:

```
Where<Account, String> where = queryBuilder.where();
where.or(
  where.and(
    where.eq(Account.NAME_FIELD_NAME, "foo"),
    where.eq(Account.PASSWORD_FIELD_NAME, "_secret")),
  where.and(
    where.eq(Account.NAME_FIELD_NAME, "bar"),
    where.eq(Account.PASSWORD_FIELD_NAME, "qwerty")));
```

This produces the following approximate SQL:

```
SELECT * FROM account
  WHERE ((name = 'foo' AND password = '_secret')
         OR (name = 'bar' AND password = 'qwerty'))
```

If you want to do complex queries linearly, you can even use Reverse Polish Notation (of all things). There is a `Where.or(int)` and `Where.and(int)` methods which do the operation on the previous number of specified clauses.

```
where.eq(Account.NAME_FIELD_NAME, "foo");
where.eq(Account.PASSWORD_FIELD_NAME, "_secret");
// this does an AND between the previous 2 clauses
// it also puts a clause back on the stack
where.and(2);
where.eq(Account.NAME_FIELD_NAME, "bar"),
where.eq(Account.PASSWORD_FIELD_NAME, "qwerty")));
// this does an AND between the previous 2 clauses
// it also puts a clause back on the stack
where.and(2);
// this does an OR between the previous 2 AND clauses
where.or(2);
```

The `QueryBuilder` also allows you to set what specific select columns you want returned, specify the 'ORDER BY' and 'GROUP BY' fields, and various other SQL features (LIKE, IN, >, >=, <, <=, <>, IS NULL, DISTINCT, …). See section Where Capabilities. You can also see the javadocs on `QueryBuilder` and `Where` classes for more information. Here's a good SQL reference site.