Writing to Files

Simone Alessandria
AUTHOR, TRAINER AND PROUD DEVELOPER

www.softwarehouse.it

Overview

Using Files

- path_provider
- Dart:io

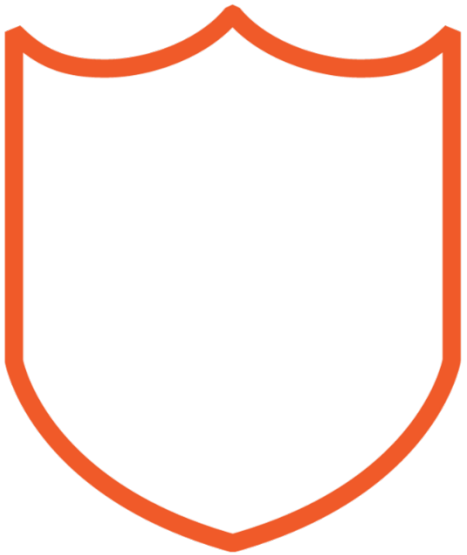
Sharing Files

- share

Moor

Moor is a reactive persistence library for Dart and Flutter applications.

Moor Features



Type Safety

Stream Queries

Data Validation

Less Code

```dart
part 'moor_database.g.dart';

class BlogPosts extends Table {

IntColumn get id =>
integer().autoIncrement().call();

RealColumn get someFloat => real()();

TextColumn get name =>
text().withLength(min: 1, max: 80)();

BlobColumn get image =>
blob()();

DateTimeColumn get date =>
dateTime().nullable()();

BoolColumn get isImportant =>
boolean().withDefault(Constant(false))
();

}
```

- part statement
- Table class
- At the end of the Column definition you add a call() method
- You can also add () instead of .call()
- Define field properties: withLength, nullable, withDefault, autoIncrement...
- DateTime is saved as a Unix TimeStamp
- Booleans are saved as integers

```
Future<List<BlogPost>> getPosts() =>
select(blogPosts).get();
```

```
Stream<List<BlogPost>> getPosts() =>
select(blogPosts).watch();
```

- Returns a List of objects (Future)
- Returns a List of objects (Stream)

```
Future<int> insertPost(BlogPost
post) => into(blogPosts).insert(post);
```

- Insert

```
Future<bool> updatePost(BlogPost
post) =>
update(blogPosts).replace(post);
```

- Update

```
Future<int> deletePost(BlogPost
post) =>
delete(blogPosts).delete(post);
```

- Delete

Summary

Moor

- Create, Read, Update, Delete
- Future and Stream

Code Generation

- part

Provider