LiteDB

FileStorage

To keep its memory profile slim, LiteDB limits the size of a documents to 1MB. For most documents, this is plenty. However, 1MB is too small for a useful file storage. For this reason, LiteDB implements `FileStorage`, a custom collection to store files and streams.

`FileStorage` uses two special collections:

- The first collection stores file references and metadata only (by default it is called `_files`)

```
{
    _id: "my-photo",
    filename: "my-photo.jpg",
    mimeType: "image/jpg",
    length: { $numberLong: "2340000" },
        chunks: 9,
    uploadDate: { $date: "2020-01-01T00:00:00Z" },
    metadata: { "key1": "value1", "key2": "value2" }
}
```

- The second collection stores binary data in 255kB chunks (by default it is called `_chunks`)

```
{
    _id: { "f": "my-photo", "n": 0 },
    data: { $binary: "VHlwZSAob3Igc ... GUpIGhlcmUuLi4" }
}
{
    _id: { "f": "my-photo", "n": 1 },
    data: { $binary: "pGaGhlcmUuLi4 ... VHlwZSAob3Igc" }
}
{
   ...
}
```

Files are identified by an `_id` string value, with following rules:

- Starts with a letter, number, `_` , `-` , `$` , `@` , `!` , `+` , `%` , `;` or `.`
- If contains a `/` , must be sequence with chars above

To better organize many files, you can use `_id` as a `directory/file_id` . This will be a great solution to quickly find all files in a directory using the `Find` method.

Example: `$/photos/2014/picture-01.jpg`

The `FileStorage` collection contains simple methods like:

- **`Upload`** : Send file or stream to database. Can be used with file or `Stream` . If file already exists, file content is overwritten.
- **`Download`** : Get your file from database and copy to `Stream` parameter
- **`Delete`** : Delete a file reference and all data chunks
- **`Find`** : Find one or many files in `_files` collection. Returns `LiteFileInfo` class, that can be download data after.
- **`SetMetadata`** : Update stored file metadata. This method doesn't change the value of the stored file. It updates the value of `_files.metadata` .
- **`OpenRead`** : Find file by `_id` and returns a `LiteFileStream` to read file content as stream

```
// Gets a FileStorage with the default collections
var fs = db.FileStorage;

// Gets a FileStorage with custom collection name
var fs = db.GetStorage<string>("myFiles", "myChunks");

// Upload a file from file system
fs.Upload("$/photos/2014/picture-01.jpg", @"C:\Temp\picture-01.jpg");

// Upload a file from a Stream
fs.Upload("$/photos/2014/picture-01.jpg", "picture-01.jpg", stream);
```

```
// Find file reference only - returns null if not found
LiteFileInfo file = fs.FindById("$/photos/2014/picture-01.jpg");

// Now, load binary data and save to file system
file.SaveAs(@"C:\Temp\new-picture.jpg");

// Or get binary data as Stream and copy to another Stream
file.CopyTo(Response.OutputStream);

// Find all files references in a "directory"
var files = fs.Find("$/photos/2014/");
```

`FileStorage` does not support transactions to avoid putting all of the file in memory before storing it on disk. Transactions *are* used per chunk. Each uploaded chunk is committed in a single transaction.

Made with ♥ by LiteDB team - @mbdavid - MIT License