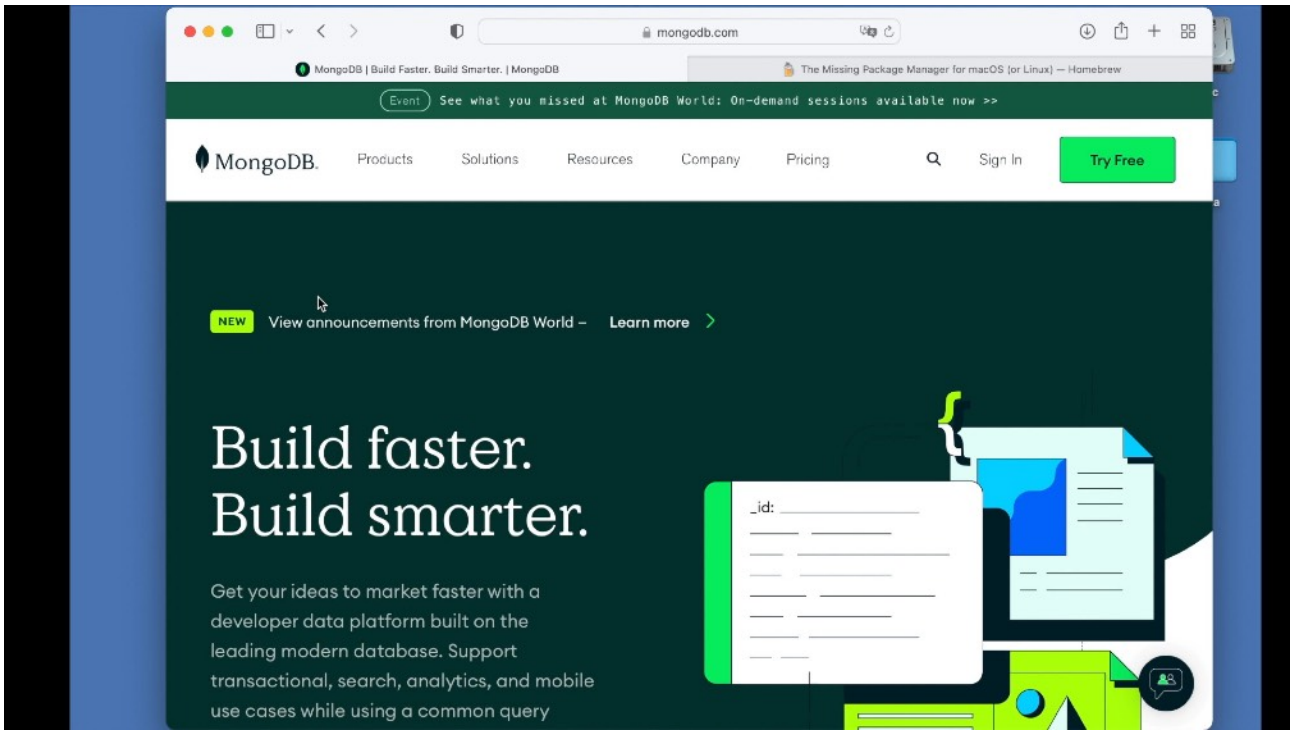
Store containers in a MongoDB



Recently a client tried our MongoDB functions in MBS FileMaker Plugin. They asked us how to store binary data in the MongoDB in an efficient way. For example that may be a picture in a container field in their FileMaker database. While we can store Base64 or Hex encoded data as text in a text field, we prefer to store it as binary data to save space.

MongoDB has a binary data typein BSON to store various things. This includes several sub types like the binary version of UUID, MD5 hashes, encrypted data, compressed time series data or generic binary data. The UUID version for example uses only 16 bytes instead of 36 characters for the text representation.

We can use the binary version by passing a JSON object as value which contains $binary and that has a JSON object with two keys: subType and base64. We can create such a JSON easily in FileMaker with either the built-in JSON functions or the MBS functions. The sub type is just a decimal to specify the type and we pass zero for this value. The base64 field receives the base64 encoded data of our image. We pass it in the JSON base64, but the MongoDB will store it in binary way, so our encoding here is just for the transfer between FileMaker and the MBS FileMaker Plugin. It will go over the network connection in the binary form.

Let's add a record with the following script. First we build the json as required:

Set Variable [$data ; Value: JSONSetElement ("{}" ; ["subType"; 0; JSONString]; ["base64"; Base64Encode (test::Image); JSONString])]
Set Variable [$binary ; Value: JSONSetElement ("{}" ; ["$binary"; $data; JSONRaw])]
Set Variable [$json ; Value: JSONSetElement ("{}" ; ["FileID" ; 123 ; JSONNumber] ; ["filename" ; "test.png" ; JSONString] ; ["Data" ; $binary ; JSONRaw])]

The JSON looks like this:

{"Data":{"$binary":
{"base64":"iVBORw0K...ElFTkSuQmCC\r\n","subType":0}},"FileID":123
,"filename":"test.png"}

We can insert it into the current collection over a connection to a MongoDB database:
Set Variable [$r ; Value: MBS("MongoDB.InsertOne"; $Mongo; $json)]
Show Custom Dialog ["Result" ; $r]

After we stored the value, let's find it again. For that we use the FileID parameter we put in the record for a find operation:

```
Set Variable [ $r ; Value: MBS( "MongoDB.Find"; $Mongo; "{\"FileID\": 123}") ]
If [ MBS("ISError") ]
  Show Custom Dialog [ "Failed to find" ; $r ]
Else
  Set Variable [ $json ; Value: MBS( "MongoDB.CursorNext"; $Mongo) ]
  If [ Length ( $json ) > 0 ]
    Set Field [ test::Insert JSON ; $json ]
    Set Field [ test::Image ; Base64Decode ( JSONGetElement
( test::Insert JSON ; "Data.$binary.base64" ); JSONGetElement
( test::Insert JSON ; "filename" ) ) ]
  End If
End If
```

Once we got a cursor for the result, we can ask for the first record and unpack the binary data. Works nice in our tests, so please try it and please do not hesitate to contact us with your questions.