Multimedia im Netz (Online Multimedia)

Wintersemester 2014/15

Übung 08 (Hauptfach)

Today‘s Agenda

- Quiz Preparation
- NodeJS & MongoDB
  - What is it?
  - Terminonolgy
  - Setup
  - Usage examples
  - Middleware: Monk

Flashback!

- Let’s take 10 minutes and prepare an MMN quiz!
- Take out lecture/tutorial slides and scan for familiar content.
- Don’t hesitate, just say any question that pops into your brain

MongoDB

- “NoSQL”
  - No SQL
  - Not only SQL
- No “Schema”: Code defines Schema
- No complex joins
- Document Driven / Object
  - Different types of documents in the same collection
  - Deep Query ability
  - Index on any attribute
- High Scalability
- JSON Interface

Terminology

SQL	MongoDB
database	database
table	collection
row	document
column	field
index	index
table joins	embedded documents and linking
primary key	primary key
UNIQUE column	Automatically generated _id field

Install MongoDB

- Download here and run locally:
  http://www.mongodb.org/downloads
- Start daemon:
  ```
  $ /path/to/your/mongo/installation/bin/mongod
  ```
- Launch mongo:
  ```
  $ mongo
  ```
- Create a database:
  ```
  $ use mmn1415
  ```
- Verify:
  ```
  $ show dbs
  ```

For today's tutorial

- MongoDB is not available at the CIP Pool ☹
- Those of you who do not have their own laptop can easily follow along on http://try.mongodb.org/

Creating Collections

- Collections are created implicitly (as are databases)
- Alternative:
  `db.createCollection("collectionName")`

SQL	MongoDB

CREATE TABLE users (id MEDIUMINT
NOT NULL AUTO_INCREMENT,
user_id Varchar(30), age Number,
status char(1),
PRIMARY KEY (id))
	db.users.insert(
	{
	user_id: "abc123",
	age: 55,
	status: "A"
	}
	)

Inserting Data

- Inserts are JSON Objects
- Multiple objects can be wrapped into an array and then inserted

SQL	MongoDB
INSERT INTO users (user_id, age, status) **VALUES** ("bcd001", 45, "A")	**db.users.insert(** { user_id: "bcd001", age: 45, status: "A" })

Multiple types in the same collection

- ```
  db.collection.insert({foo:'bar'});
  db.collection.insert({lorem:‘ipsum'});
  ```

- Developers have to take care about what objects to put in a collection

- Yet, Mongo is really flexible!
  ```

Querying

SQL	MongoDB
SELECT * FROM users	db.users.find()
SELECT id, user_id, status FROM users	db.users.find({ }, { user_id: 1, status: 1 })
SELECT user_id, status FROM users	db.users.find({ }, { user_id: 1, status: 1, _id: 0 })
SELECT * FROM users WHERE status = "A"	db.users.find({ status: "A" })
SELECT user_id, status FROM users WHERE status = "A"	db.users.find({ status: "A" }, { user_id: 1, status: 1, _id: 0 })
SELECT * FROM users WHERE status != "A"	db.users.find({ status: { $ne: "A" } })

http://docs.mongodb.org/manual/reference/sql-comparison/

Update & Delete

SQL	MongoDB
UPDATE users **SET** status = "C" **WHERE** age > 25	**db.users.update**({ **age**: { **$gt**: 25 } }, { **$set**: { **status**: "C" } }, { **multi**: **true** })
UPDATE users **SET** age = age + 3 **WHERE** status = "A"	**db.users.update**({ **status**: "A" } , { **$inc**: { **age**: 3 } }, { **multi**: **true** })

SQL	MongoDB
DELETE FROM users **WHERE** status = "D"	**db.users.remove**({ **status**: "D" })
DELETE FROM users	**db.users.remove**({})

Operators

- Operators are special keys inside queries in MongoDB
- You can't write 'someKey' != 'someValue'.
- Most common operators:
  - $ne, $gt, $lt, $gte, $lte
  - $and, $or
- Example:

SQL	MongoDB
SELECT * **FROM** users **WHERE** status = "A" **OR** age = 50	**db.users.find(** { **$or**: [{ **status**: "A" } , { **age**: 50 }] })

http://docs.mongodb.org/manual/reference/operator/

NodeJS and MongoDB

- There are a couple of implementations for NoSQL/MongoDB middleware in NodeJS
- For MongoDB, the most prevalent examples are
  - monk
  - mongoose
- In the tutorial, we use monk, but mongoose works quite similar.

Using MongoDB middleware

- ```
  var db = require('monk')('localhost/mmn-1415-assignment08');
  ```

- This has to come very early in the middleware chain:

```
app.use(function(req,res,next){
    req.db = db;
    next();
});
```
```

Accessing the database with monk

```
var express = require('express');
var router = express.Router();
router.get('/users',function(req,res){
  var db = req.db;
  var users = db.get('users');
  users.find({},{},function(e,docs){
    res.json(docs);
  });
});
module.exports = router;
```

Note: it's not necessary to require monk here! Why?

Monk - Operations

- Monk wraps statements to JS Functions:
  - find()
  - findOne()
  - update()
  - updateById()
- All (!) queries are asynchronous! Callbacks!

```
router.get('/users',function(req,res){
    var db = req.db;
    var users = db.get('users');
    users.find({status : {$ne : 'A'}},function(e,docs){
        res.json(docs);
    });
});
```

Further Tips & Tricks: Express Generator

- Express Apps can be easily created with the express generator:
  $ npm install -g express-generator
  $ cd /your/path/
  $ express myNewApp && cd myNewApp

Express Generator Output

```
create : myNewApp
  create : myNewApp/package.json
  create : myNewApp/app.js
  create : myNewApp/public
  create : myNewApp/public/stylesheets
  create : myNewApp/public/stylesheets/style.css
  create : myNewApp/public/images
  create : myNewApp/routes
  create : myNewApp/routes/index.js
  create : myNewApp/routes/users.js
  create : myNewApp/views
  create : myNewApp/views/index.jade
  create : myNewApp/views/layout.jade
  create : myNewApp/views/error.jade
  create : myNewApp/bin
  create : myNewApp/bin/www
  create : myNewApp/public/javascripts

  install dependencies:
    $ cd myNewApp && npm install

  run the app:
    $ DEBUG=myNewApp ./bin/www
```

Further Tips & Tricks

- MongoDB does not allow you have keys containing dots, eg:
  `db.users.insert({'127.0.0.1': 'up'}); // not allowed`
- Solution: Create a unique hash of the key and store it.

```
String.prototype.hashCode = function() {
    var hash = 0, i, chr, len;
    if (this.length == 0) return hash;
    for (i = 0, len = this.length; i < len; i++) {
        chr   = this.charCodeAt(i);
        hash  = ((hash << 5) - hash) + chr;
        hash |= 0; // Convert to 32bit integer
    }
    return hash;
};
```

Assignment 08

- **Topic: Extending the Feedback Voting App & Theory**
- **Due in:** 2 Weeks
- **Due date:** 22.12.2014
- Notes:
  - A solution for assignment 07 will be uploaded after the deadline
  - You can (but don't have to) use this solution as a basis for assignment 08
  - Tip: Prepare the theoretic questions already this week, the programming task is fairly easy then.

Programming Consultation

- NodeJS, Express, MongoDB – tough cookies?
- Come by on Wednesday, 10.12.2014 at 18:00h and ask your questions.
- Please sign up first:

  http://bit.ly/mmn-1415-pb02

Thanks!

What are your questions?