MongoDB and Databases

Time to dump your JSON somewhere

Also HUGE thanks to Plexus for sponsoring the IEEE Fall Mini-Projects

Databases

- Way to save data in persistent memory
- Can host on server
  - Allows to not physically hold data
- Can host on local computer
  - Or in this case the Pi
- No more saving and reading to TXT Files

Every Database is CRUD

- **C**reate
  - Add new data to database
- **R**ead
  - Get data from database
- **U**pdate
  - Change data from database
- **D**elete
  - Erase data from database

API vs Database

- API is the **Application** that does the CRUD
- Example
  - https://api.github.com/users/sjfricke
  - Fetches JSON data from a Database
  - The backend server code is the API


```
{ "login": "sjfricke",
  "id": 9061055,
  "avatar_url": "https://avatars.githubusercontent.com/u/9061055?v=3",
  "html_url": "https://github.com/sjfricke,
  "email": "sjfricke@wisc.edu
  "updated_at": "2016-10-08T05:48:35Z" }
```

Relational Databases

- Tables
- SQL
  - The Language to **query** from table
- MySQL, SQL Server, PostgreSQL, Oracle, etc…
  - Relational Database Management System
  - Use their own “dialect” of SQL
- CS 564

SQL Table Example

- http://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all
- SELECT * FROM Customers;
- SELECT City, ContactName FROM Customers WHERE Country='Mexico';
- UPDATE Customers SET ContactName='Alfred Schmidt', City='Hamburg‘ WHERE CustomerName='Alfreds Futterkiste';
- SELECT Customers.CustomerName, Orders.OrderID FROM Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName;

Non-Relational

- No More Tables! (ノ°□°)ノ ⌒ ┻━┻
- Also called “NoSQL”
- Can scale “easier” then SQL
  - Have you broke a SQL Schema before… it’s not fun
- Web 2.0 choice of database
  - Facebook, Google, Amazon, etc.
    - **NOTE**: They built their own databases and search function to maximize search time… but MongoDB is good enough for people who don’t have billions of users

What is MongoDB

- A “NoSQL” Database
- Non-Relational
- Free
- Not default on Raspberry Pi’s
  - I added it for everyone
    - PDF on the GitHub how to do it
- “A big JSON garbage bin”
  - Only if you are not careful

JSON Overview

- Open your browser of choice
  - Only if that choice is Google Chrome
    - Windows ---------- Ctrl+Shift+J
    - Macs ---------- Option+Command+J
- Go to Lesson_2 in GitHub folder for practice.json
- https://github.com/sjfricke/IEEE_RaspberryPi_Socket_Pokemon/blob/master/Tutorial/Lesson_2/practice.json

- Select all (ctrl+a) and copy it (ctrl+c)
- Type: **var data =**
  - Then paste (ctrl+v) data on screen and hit enter

- Undefinded?
  - Type: **data**

NOTE: There is never a comma after last listed item

- Can use built-in JavaScript
  <ArrayName>.find(function(e){return e.name == "Alakazam"})

Mongod vs Mongo ?

- **Mongod** is “Running MongoDB”
  - Stands for “Mongo Daemon”
- **Mongo** is the command line shell interface which we config with
  - To access it just type “mongo”

- Other options include **mongoimport** and **mongoexport** (will use later)

Linux Services

- Check your Mongo Service by typing:
  sudo Serivce mongodb status
- Other operations

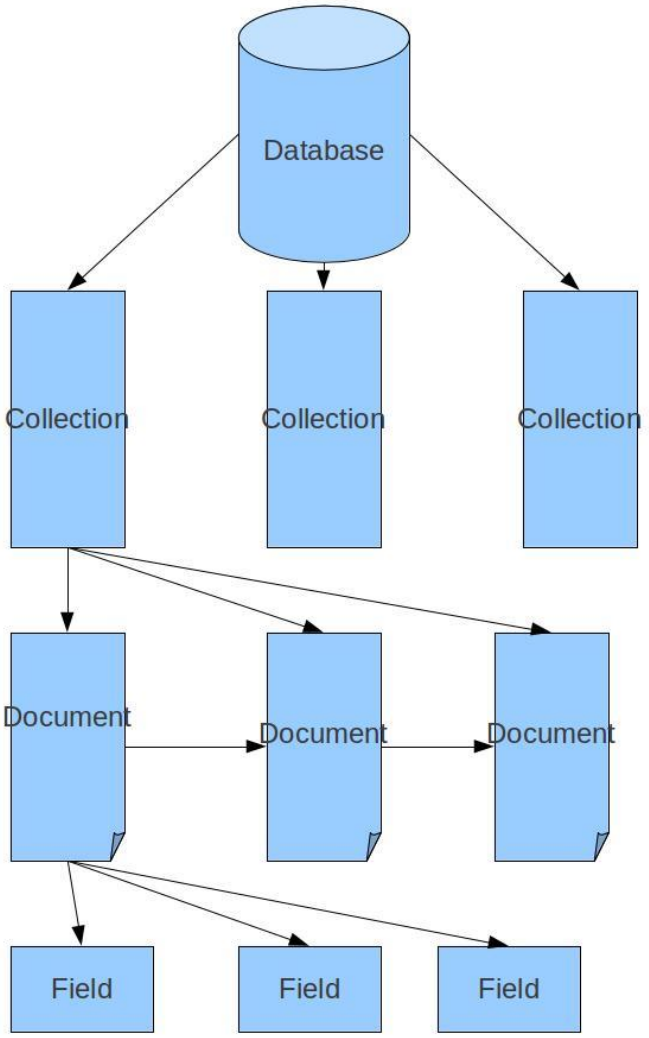
  sudo Serivce mongodb **start**

  sudo Serivce mongodb **stop**

  sudo Serivce mongodb **restart**

Mongo

- Database
  - Using only 1
- Collections
  - One big JSON value
- Documents
  - Each part of the JSON
- Field
  - The data you want



- ***Database***: CollegeOfEngineering
  - ***Collection***: Professors
  - ***Collection***: Buildings
  - ***Collection***: Students
    - ***Documents***:
      - { Name : “Spencer”, Age : 21, Type : “ECE” }
      - { Name : “Fred”, Age : 24, Type : “ME” }
      - { Name : “Willy”, Age : 19, Type : “BME” }

Name	Age	Type
Spencer	21	ECE
Fred	24	ME
Willy	19	BME

      - » ***Field***: “Spencer”
      - » ***Field***: 24

Mongo Shell

- Start by setting the Database we are going to use
  - Lets call it **Students**
  - Type: use Students
  - Sets variable "**db"** to the database
- To see all Databases
  - Type: show dbs

- Will not create one until written too

Add some data

Name	Age	Type
Spencer	21	ECE
Fred	24	ME
Willy	19	BME

- Will put in Collection **Engineers**
  - db.Engineers.insert(
    {
      Name : “<insertYourName>” ,
      Age : “<insertYourAge>”,
      Type : “<insertYourType>”
    }
)

NOTE: No comma after your Type

Other ways to insert

- Set a variable
  - student2 = { Name : “Fred” }
  - db.Engineers.insert(student2)
  - HINT: Tab completes still works in Mongo shell

- What happens if you do the insert twice?

Read your data

- Use differnet “Find” methods to query data
  - db.Engineers.find()
  - db.Engineers.find({Name : “Fred”})
  - db.Engineers.findOne({Name : “Fred”})
- Sort it as well
  - db.Engineers.find().sort({Name: 1 })
    - Use -1 for descending order
- Count
  - db.Engineers.count()
  - db.Engineers.find({Name : “Fred”}).count()

Messed up? Just Update!

- Update() takes 3 parameters
  - Query to find the document you want
  - What to update with it
  - Additional options (optionial parameter)

- First we will add information
  - db.Engineering.update(
    { Name : ”Fred” },
    {
    $set : { Age : 23, Type: “ME” }
    }
)
- Works the same to edit information
  - db.Engineering.update(
    { Name : ”Fred” },
    {
    $set : { Age : 24 }
    }
)

Need More Data!

- Can take raw JSON objects and import them
  - Can also export collections to JSON files
  - Perform **outside** the mongo shell
- mongoimport --db World --collection Population --drop --file Population.json
- mongoimport --db World --collection NobelPrize --drop --file NobelPrize.json
- mongoimport --db World --collection Companies --drop --file Companies.json
  - Save time and use the ./importScript

Queries

- Don't forget to use World
- show collections
- db.Population.find({ males : { $gt : 2250000 } })
- To much information?
  - Add 2nd parameter to find to limit fields
  - Uses 1 and 0 to turn on or off
    - _id will be set to 1 by default
- db.Population.find(

  { males : { $gt : 2250000 } },

  { males : 1, age : 1, _id : 0 }

)

Find the order of the most females in the 30-39 age range

- Easy!
- db.Population.find({ age : {
  $gte : 30, $lte : 39
  }
  },
  {age : 1, females : 1, _id:0}
).sort(
  { females : -1 }
)

When has there been only two Nobel Prize Laureates and one of them was named either John, Bob, and/or Eric?

- Oddly specific, talk about being bored
- db.NobelPrize.find(

```
    {
            "laureates.firstname" : {
                    $in: ["John", "Bob", "Eric"]
             },
             laureates : {$size: 2}
    },
    {year : 1, category:1, _id:0}
)
```

Moral of the story

- Use a GUI if possible
- My choice is MongoChef
  - Note: Free for non commerical use
  - Has built in import and export functions as well