Building an IGT Collider

[My notes on how to build the environment to collect, manage and analyze data for studies related to Institutional Grammar Theory]

Contents

[Planning the Project 4](#_Toc44990802)

[Learning the Tools 4](#_Toc44990803)

[Setting up the Environment 4](#_Toc44990804)

[Using NODEJS, Express, Sequelize, and MySQL. 4](#_Toc44990805)

[Created new folder 4](#_Toc44990806)

[Install Dependencies 4](#_Toc44990807)

[Add a dev dependency 5](#_Toc44990808)

[Add Start and Dev scrips 5](#_Toc44990809)

[Create app.js 5](#_Toc44990810)

[Initializing a database connection 5](#_Toc44990811)

[NodeJS 6](#_Toc44990812)

[Express 6](#_Toc44990813)

[Routing 6](#_Toc44990814)

[Defining a Route 7](#_Toc44990815)

[Launching the code 7](#_Toc44990816)

[Chapter 1 – The Beginning 7](#_Toc44990817)

[Access 7](#_Toc44990818)

[Data Base Connections 7](#_Toc44990819)

[MySQL Connector/NodeJS 7](#_Toc44990820)

[R Connections 8](#_Toc44990821)

[Data Models 8](#_Toc44990822)

[Data Dictionaries 8](#_Toc44990823)

[Debugging 8](#_Toc44990824)

[Environments 8](#_Toc44990825)

[Integrated Development Environment (IDE) 8](#_Toc44990826)

[Files 8](#_Toc44990827)

[Directory 8](#_Toc44990828)

[Open File – 8](#_Toc44990829)

[Manipulating File Names 8](#_Toc44990830)

[Import Excel files 9](#_Toc44990831)

[Write Comma Delimited (CSV) 9](#_Toc44990832)

[Functional Programming 9](#_Toc44990833)

[GIT 9](#_Toc44990834)

[Files 9](#_Toc44990835)

[Installing and Maintaining NodeJS 9](#_Toc44990836)

[Introduction 9](#_Toc44990837)

[Java Script 10](#_Toc44990838)

[Mysql 10](#_Toc44990839)

[NLTK 10](#_Toc44990840)

[Node JS 11](#_Toc44990841)

[Documentation 11](#_Toc44990842)

[Tutorial 11](#_Toc44990843)

[Package Management 11](#_Toc44990844)

[Sequelize 11](#_Toc44990845)

[Style Guide 11](#_Toc44990846)

[Tables 11](#_Toc44990847)

[Word 12](#_Toc44990848)

[Writing Tables to Word 12](#_Toc44990849)

[Project To Do 13](#_Toc44990850)

[References 14](#_Toc44990851)

[Books 14](#_Toc44990852)

[Articles 14](#_Toc44990853)

[Manual 14](#_Toc44990854)

[Tutorials 14](#_Toc44990855)

[Websites 14](#_Toc44990856)

# Planning the Project

Before we get into acquiring and processing data, writing code, etc – we need to design our project, create a project plan. – Lecture 76 of the JavaScript tutorial is a nice motivator for this.

Step 1. Identify the fundamental things the project is to accomplish

Step 2. Structure the project using a modular concept – logical parts that interact to produce the final product.

Step 3. Database model – define the tables and links that creates the model (explain why as you build it)

# Learning the Tools

I found a variety of places to learn how to use each of the tools in this project.

Youtube video on Node.js, Express, Mysql, Sequalize

# Setting up the Environment

## Using NODEJS, Express, Sequelize, Handlebars, and MySQL.

(Switching to VS Code for my IDE… July 25, 2020).

### Created new folder

– IGT within C:\Users\Jflowers6\Dropbox\NodeJS\IGT

Went into terminal – NPM Init

Package name: IGT

Version: (1.0.0)

Description: Data Entry to parse legislation observations

Entry point: app.js

Test command: [runs a test script, if provided – will skip for now]

Git repository: [will skip for now as well]

[took all other defaults for now]

### Install Dependencies

Dependencies are packages that the code relies on to function. Examples include express, handlebars, atc.

Npm I express body-parser sequelize pg pg-hstore mysql2 express-handlebars

[ran into errno 4058 – reinstalled npm

Npm install –g npm

Npm install

And everything was fine – might want to try just the npm install next time]

### Add a dev dependency

Npm I –D nodemon

### Add Start and Dev scrips

Going back into package.json – added

"start": "node app.js",  
"dev" : "nodemone app.js"

When you type “npm run dev” – starts our server with nodemon (explain nodemon later)

Browser address -> localhost:port

### Create app.js

Within the IGT directory (you should see the package.json located there)

Create a router in folder Routes – with script called obs.js

### Initializing a database connection

This is done with a js file within the config folder which I call database.js. This file initializes Sequelize, the ORM middleware for this project. The file has the following content:

// initialize database connection  
  
const Sequelize = require('sequelize');  
module.exports = new Sequelize('mydb','jim','mydev$2021',{  
 host:'localhost',  
 dialect : 'mysql',  
 operatorsAliases :'',  
  
 pool: {  
 max:100,  
 min: 0,  
 acquire: 30000,  
 idle: 10000  
 },  
});

My database model exists within MySql.

## Sequelize

Sequelize is an ORM. An ORM is an Object-Relation-Mapper. An ORM maps the language functions of, say, Java to SQL or whatever query language your database manager employs. In academic terms, an ORM ‘abstracts’ the database manager – so you can switch the backend manager without worrying about re-writing your queries to match the manager[[1]](#footnote-1).

Sequelize equates a Model with a Table in the database. .

You create a model for sequelize – creates a class to be referenced in java code. The model reflects what is stored in the database table.

#### NOTES

When you initialized Sequelize – you set up timestamps – which require columns ‘createdAt’ and ‘updatedAt’ – this caused a problem until you added the columns in the mysql schema. You can turn off the option :

const sequelize = new Sequelize(connectionURI, {

define: {

// The `timestamps` field specify whether or not the `createdAt` and `updatedAt` fields will be created.

// This was true by default, but now is false by default

timestamps: false

}

});

## NodeJS

Stylesheets – check this out <https://styled-components.com/docs/basics#installation>

## Express

Is a web application framework for Node.js released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js ([cite](https://en.wikipedia.org/wiki/Express.js))[[2]](#footnote-2)

### Routing

(summarizing the information presented at <https://expressjs.com/en/guide/routing.html> and associated pages.

Routes are how an application service requests.

A request is made to an endpoint – which is a URI (or path) and a specific HTTP request method (GET, POST, etc).

Route definition takes the following structure:

app.METHOD(PATH, HANDLER)

Where:

* app is an instance of express.
* METHOD is an [HTTP request method](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol#Request_methods), in lowercase.
* PATH is a path on the server.
* HANDLER is the function executed when the route is matched.

### Defining a Route

You define routing using methods of the Express app object that correspond to HTTP methods; for example, app.get() to handle GET requests and app.post to handle POST requests. For a full list, see [app.METHOD](https://expressjs.com/en/4x/api.html#app.METHOD). You can also use [app.all()](https://expressjs.com/en/4x/api.html#app.all) to handle all HTTP methods and [app.use()](https://expressjs.com/en/4x/api.html#app.use) to specify middleware as the callback function (See [Using middleware](https://expressjs.com/en/guide/using-middleware.html) for details).

Routes are organized under the routes folder

## Launching the code

From the project directory:

* Npm run dev
* To access via browser use address *localhost:8000*  (or whatever port number you have loaded for the app)

# Chapter 1 – The Beginning

In the beginning, there was no database model – so

In the database model

# Access

Exporting Tables and Relations to MySql

# Data Base Connections

This was useful

And, this is a good tutorial

### MySQL Connector/NodeJS

[Documentation](https://dev.mysql.com/doc/dev/connector-nodejs/8.0/)

## R Connections

# Data Models

Database models guide the project.

The data model for this project requires several tables

|  |  |  |
| --- | --- | --- |
| Table | Description | Columns |
| User | Contains the User identification fields (Name, University, etc). Used to validate a user to access the data, and to track who edited or appended to the data, and when. | ID – Unique Key  Last name  First Name  Email  University/organization  Phone  [may be changed to meet stds for user validation – see passport] |
| Document |  | Doc ID – Unique Key |
| Case | Describes the organization that owns the documents and who “owns” the docoments (e.g. President, CISO, CIO, etc). |  |
|  |  |  |

# Data Dictionaries

# Debugging

When you need to step through the code and see what is happening

# Environments

## Integrated Development Environment (IDE)

[WebStorm](https://www.jetbrains.com/webstorm/)

WebStorm [Documentation](https://www.jetbrains.com/help/webstorm/2020.1/meet-webstorm.html?utm_campaign=WS&utm_content=2020.1&utm_medium=link&utm_source=product)

# Files

## Directory

Setting word directory/project directories

## Open File –

When you want to have a window to pick you input file…

### 

## Manipulating File Names

## Import Excel files

## Write Comma Delimited (CSV)

# Functional Programming

Functional programming is a programming philosophy based on lambda calculus. Lambda calculus was created by Alonza Church. Lambda calculus consists of constructing lambda terms and performing reduction operations on them (Wikipedia).

A lambda term is a component of lambda calculus (expression, variable, abstraction). An expression can be a variable name, an abstraction, or a combination of the two.

An abstraction is a function. An abstraction is applied to an argument. An argument is an input value. The head of a function is a λ followed by a variable name (eg. λx). The body of the function follows… (e.g. λx.x) . The dot separates the parameters of the lamda function from the body.[[3]](#footnote-3)

# GIT

Literature

This documentation is a good history - [on version control](https://git-scm.com/)

Files –

Git has three main states that files occupy:

* Modified – you have changed the file but have not committed the file to the database
* Staged – a modified file marked to to into the next commit snapshot
* Committed – data (modified file) is stored in the database

# Installing and Maintaining NodeJS

* [NodeJS.org](https://nodejs.org/en/)

# Introduction

Seems you need to use node js by creating a server;

Here is the code for starting a service

const http = require('http');  
  
const hostname = '127.0.0.1';  
const port = 3000;  
  
const server = http.createServer((req, res) => {  
 res.statusCode = 200;  
 res.setHeader('Content-Type', 'text/plain');  
 res.end('Hello World');  
});  
  
server.listen(port, hostname, () => {  
 ***console***.log(`Server running at http://${hostname}:${port}/`);  
});

Place the IP in the browser and bob is your uncle.

Now you can play with the tutorial:

<https://nodejs.dev/>

# Java Script

[Tutorial](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/https:/developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basicsJavaScript_basics) – Mozillar.org

# Mysql

You need to load the mysql package using npm (npm install mysql –g)

Documentation is [here](https://github.com/mysqljs/mysql)

The query functions are included.

# NLTK

Other Research Code

Her

# Node JS

### Documentation

[Github directory](https://github.com/Microsoft/PTVS/)

Developer.mozilla.org/en-US/docs/Glossary/JavaScript – key references ([link](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript))

Node fs -- <https://nodejs.org/en/docs/>

## Tutorial

<https://nodejs.dev/>

# Package Management

NPM -- [link](https://www.npmjs.com/); [documentation](https://docs.npmjs.com/)

# Sequelize

A promise-based Node.js ORM – middleware for database transactions

* [Site](https://sequelize.org/v5/index.html)
* [API Reference](https://sequelize.org/v5/identifiers)

# Style Guide

Google [JS style guide](https://google.github.io/styleguide/jsguide.html)

# Tables

Create

# Word

## Writing Tables to Word

# Project To Do

|  |  |  |
| --- | --- | --- |
| **Item** | **Description** | **Status** |
| Security | Will need someone to inspect the security of files, user validation etc | Incomplete |
|  |  |  |
|  |  |  |
|  |  |  |

# References

## Books

You Don’t Know JS – Sympson, Kyle

[Repository](https://github.com/jimflowers/You-Dont-Know-JS) in your Github Account

## Articles

## Manual

## Tutorials

[MySql with Node.js](https://www.youtube.com/watch?v=EN6Dx22cPRI) - traversky

## Websites

1. See “What is an ORM and Why You Should Use It” – found at <https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a>, last accessed 3 June 2020. [↑](#footnote-ref-1)
2. Found at <https://en.wikipedia.org/wiki/Express.js>, last observed 9 July 2020. [↑](#footnote-ref-2)
3. From <http://joycse06.github.io/blog/2016/07/intro-to-lambda-calculus-and-its-relation-to-functional-programming/> last accessed 20-Dec-18 [↑](#footnote-ref-3)