Node-API-Knex Guide

Part 1 Instructions

In this demo, We would be setting up a Node project with Postgres database and Knex query builder. In this demo, we will demonstrate how to perform database migration and seeding dump data. We'll create two routes, as shown below.

- `GET: / todo | Get all tasks`
- `POST: /todo | Create new task`

Part 2 How to run

1. Set up Express Server

Follow guide on https://expressjs.com/en/starter/installing.html to

- init the npm project: npm init
- install the express: npm install express
- set up the app.js helloword

app.js:

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
```

```
  console.log(`Example app listening on port ${port}`)
})
```

2. Install Knex

```
npm install knex pg
```

3. Set up knexfile.js

https://knexjs.org/guide/migrations.html#knexfile-js

```
// Holds the Database connection config
module.exports = {
    client: 'pg',
    connection: {
        host: "localhost",
        port: 5432,
        user: "postgres",
        database: "chapter4_demo",
        password: "123456",
    }
}
```

4. Migration data

Create a new migration file

```
npx knex migrate:make create_todo_table
```

5. Fill in the upgrade and downgrade function in the migration file in the migrations/ folder:

```
exports.up = function(knex) {
  // Create table TODO
  return knex.schema.createTable("todo", (table)=>{
    table.increments("id").primary()
    table.text("task", 128).notNullable()
    table.text("description", 128)
  })
};

exports.down = function(knex) {
  // Drop table TODO
  return  knex.schema.dropTableIfExists("todo")
};
```

6. Run the migration file

Migrations allow for you to define sets of schema changes so upgrading a database is a breeze. To run the migrations, you can run the command below:

```
npx knex migrate:latest
```

7. Set up the Seed Data

https://knexjs.org/guide/migrations.html#seed-cli

- Create a folder “seeds”
- Create a file seed.js

seeds/seed.js:

```
exports.seed = function(knex) {
    return knex("todo").insert([
        {task: "Task 1", description: "Task 1 description"},
        {task: "Task 2", description: "Task 2 description"},
        {task: "Task 3", description: "Task 3 description"},
    ])
```

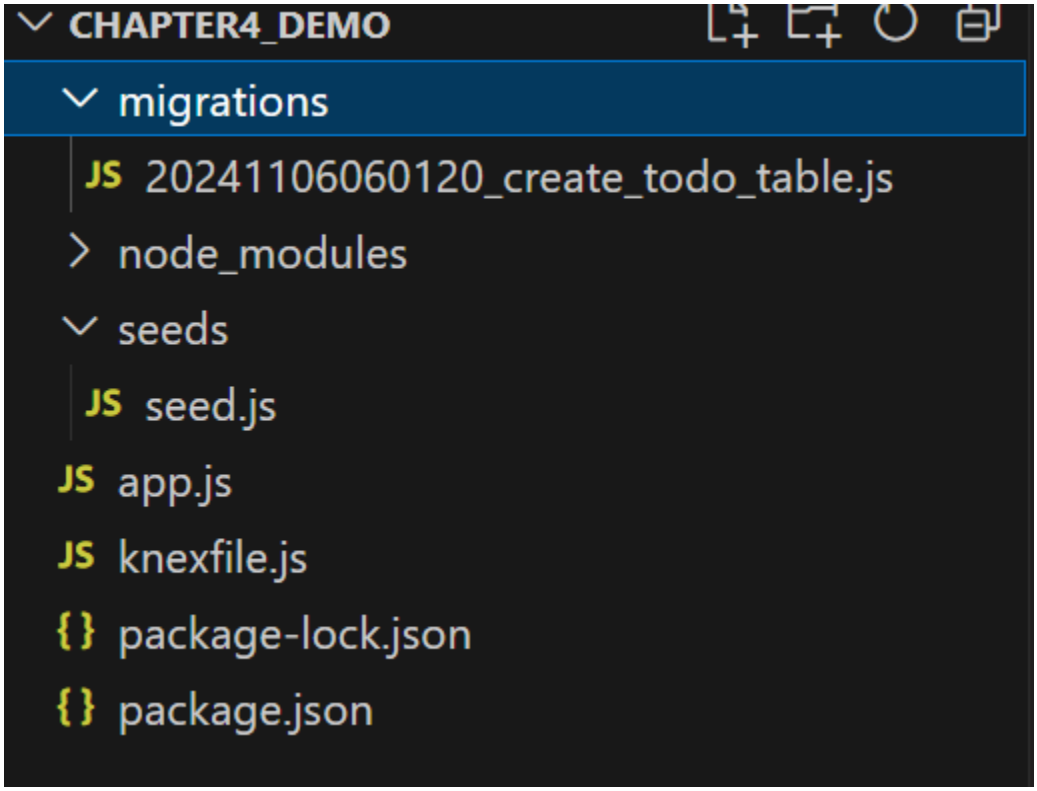
```
}
```

8. Insert the Seed Data

Seed files allow you to populate your database with test or seed data independent of your migration files. To run the seeds files you can run the command below on your terminal

```
npx knex seed:run
```

After all, your folder should look like this:



9. Start service

```
node app.js
```

Part 3 Connect DB in app.js

Import the database configuration from knexfile and start the database connection:

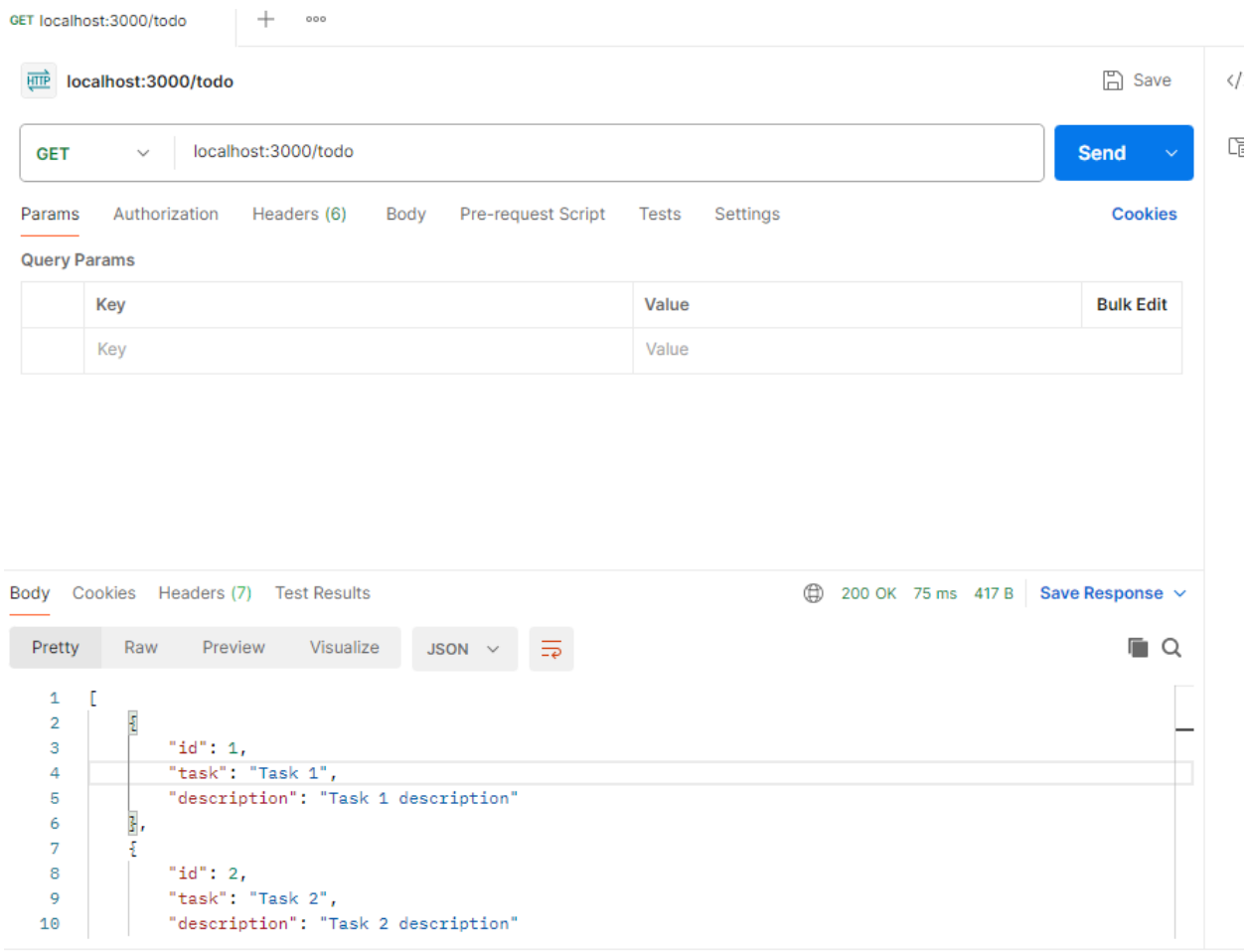
Add in app.js

```
// Import the db config from knexfile.js
const dbConfigs = require("./knexfile")
const knex = require("knex")(dbConfigs)
```

Add Get /todo API to get all todo items in the database:

```
// GET /todo
app.get('/todo', (req, res)=>{
  return knex("todo").select()
  .then((value)=>{
    return res.json(value)
  })
})
```

Use Postman to check the GET /todo API:



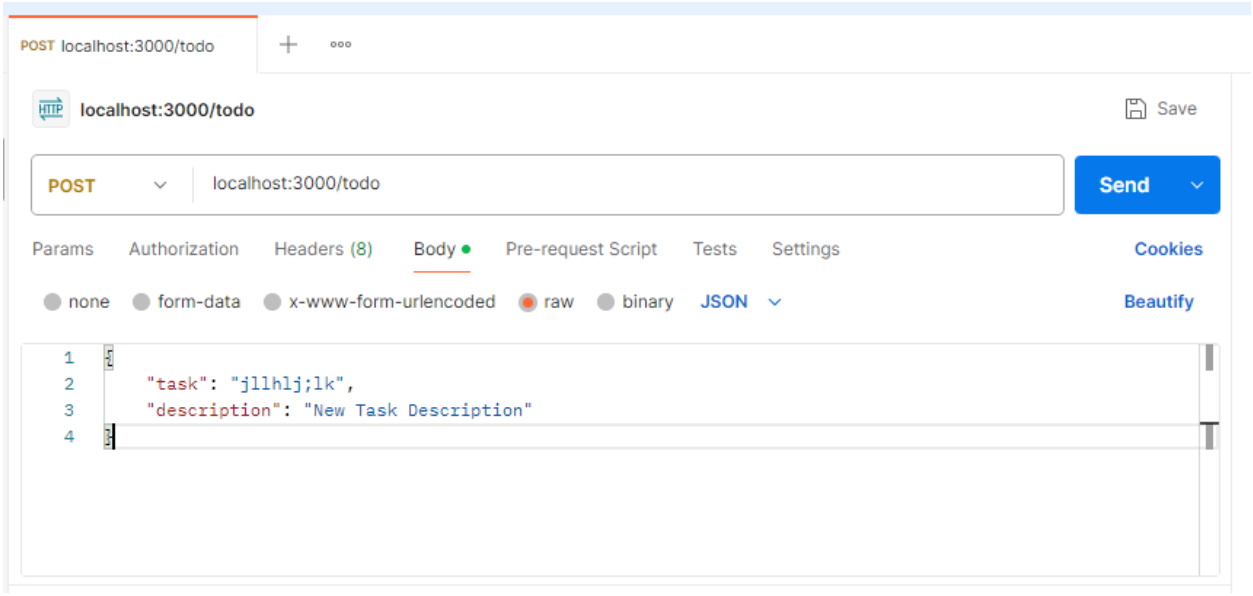
Add in app.js just below const app = express():

```
// Enable express app to parse JSON body
app.use(express.json())
```

Add Post /todo API to insert a new todo item:

```
app.post('/todo', (req, res)=>{
  const task = req.body
  return knex("todo").insert(task).then(value=>{
    res.json(value.rowCount)
  })
})
```

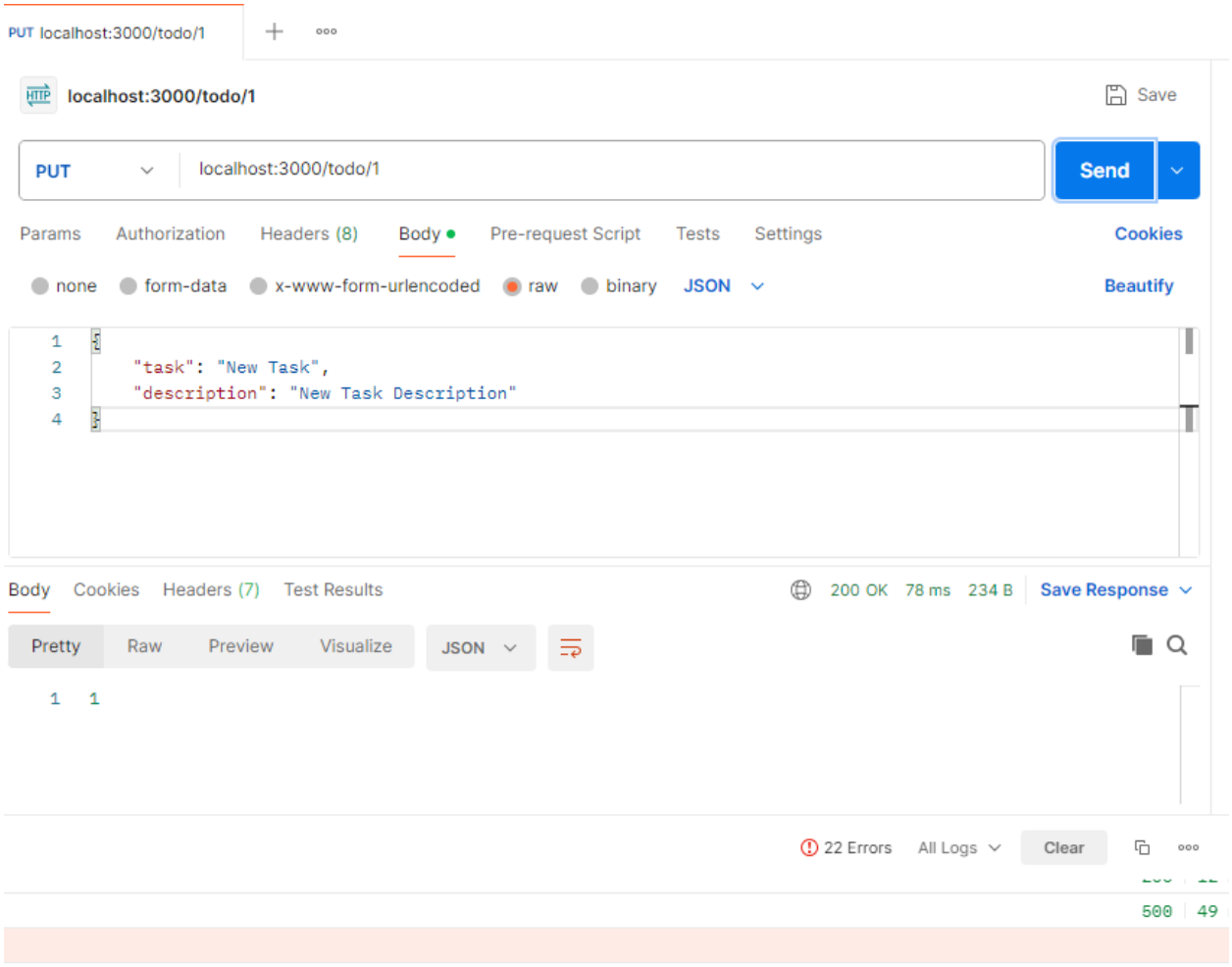
Use Postman to insert a data:



Add Put /todo/:todoId API to insert a new todo item:

```
// PUT /todo
app.put('/todo/:todoId', (req, res)=>{
  const todoId = req.params.todoId
  return knex("todo").where({
    id: todoId
  }).update({
    task: req.body.task,
    description: req.body.description,
  }).then((value)=>{
    return res.json(value)
  })
})
```

Use Postman to update a data:



Add Delete /todo/:todoId API

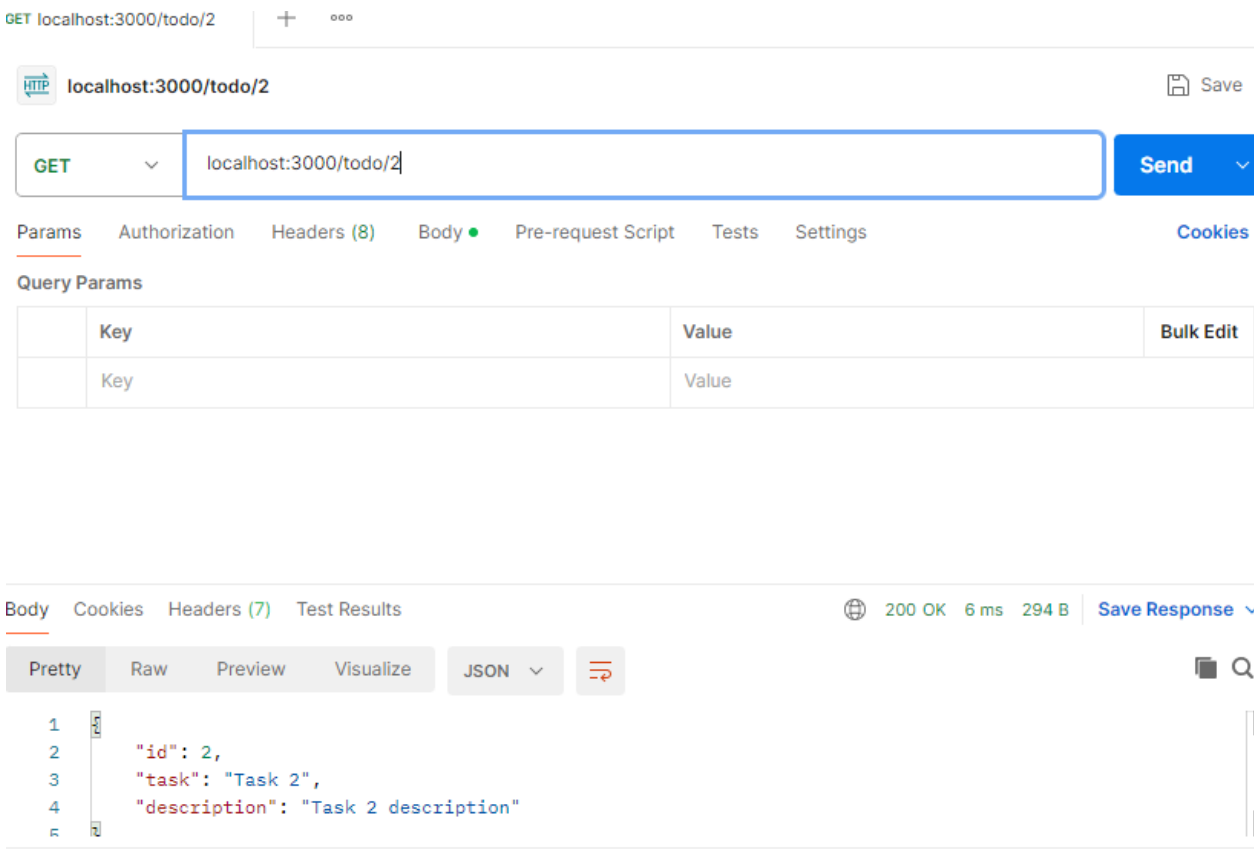
```
// Delete /todo/:todoId
app.delete('/todo/:todoId', (req, res)=>{
  const todoId = req.params.todoId
  return knex("todo").where({
    id: todoId
  }).del().then((value)=>{
    return res.json(value)
  })
})
```

Use Postman to delete a data again:

Add Get /todo/:todoId

```
// Get /todo/:todoId
app.get('/todo/:todoId', (req, res)=>{
  // Get a TODO item
  const todoId = req.params.todoId
  return knex("todo").where({
    id: todoId
  }).first().then((value)=>{
    return res.json(value)
  })
})
```

Use Postman to get a TODO item


GET localhost:3000/todo/2
localhost:3000/todo/2
Save
GET
localhost:3000/todo/2
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Bulk Edit
Key
Value

Body
Cookies
Headers (7)
Test Results
200 OK
6 ms
294 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1
2
3
4
5
{
"id": 2,
"task": "Task 2",
"description": "Task 2 description"
}