</talentlabs>

Express Lecture 11

Integrate MySQL to an Express App

Copyright © 2020 TalentLabs Limited

</talentlabs>

Agenda

- Database Connection
- Seed Files
- Integrate to an Express route
- List, Retrieve, Create, Update, Delete

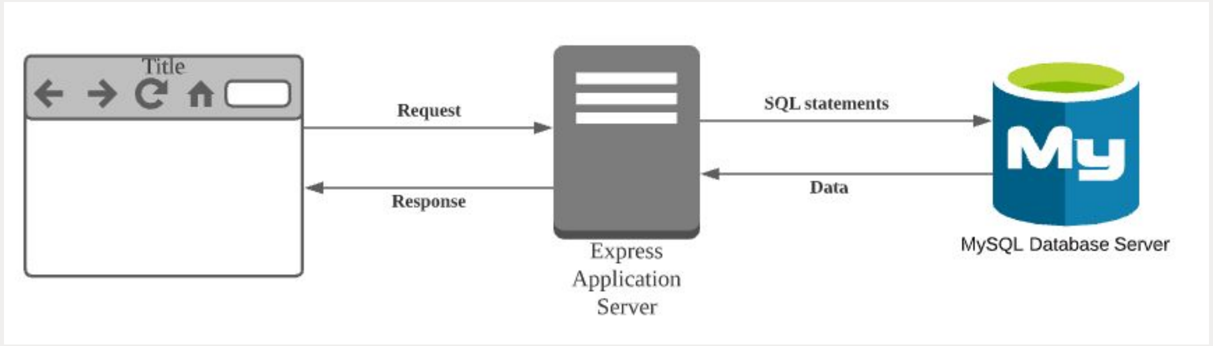
Copyright © 2020 TalentLabs Limited </talentlabs>

Database Connection

</talentlabs>

Copyright © 2021 TalentLabs Limited

Database Connection



Now, let's create a database connection for the Express Application. What is a database connection? The database connection is the **connection between our Express Application and the MySQL Database Server.**

Copyright © 2021 TalentLabs Limited

Setup Knex - to connect to the database

We will use **Knex.js** to manage our database design in Express. To install Knex we need to run:

```
npm install knex mysql --save
npx knex init
```

Next we need to tell **Knex how to connect to our database**. We need to update the database configuration in the **“knexfile.js” file**.

```
  development: {
    client: "mysql",
    connection: {
      host: "student-mysql.ccttwiegufhh.us-east-2.rds.amazonaws.com",
      user: "studentmysql",
      password: "studentmysql",
      database: "express_lecture",
    },
  },
```

Replace with your database name

Copyright © 2021 TalentLabs Limited

</talentlabs>

Seed Files

Copyright © 2021 TalentLabs Limited

What is a seed file?

- Sometimes we want to have some **initial data**. These are called seed data.
- For example, before setting up an Admin Panel for creating Products of a online store, we can have some **initial Products inserted to the database**.

In knex, there is a concept called **seed**. A seed is a file that populates the initial data into the database.

First, Let's create 2 seed files

```
npx knex seed:make initial-manufacturer
npx knex seed:make initial-product
```

Copyright © 2021 TalentLabs Limited

Default seed data file

The initial seed file contains the following content, it contains 2 steps

1. **Delete all entries** from a table.
2. Insert seed data entries.

This is **not a good start**. We don't have to delete all the data every time.

./seeds/initial-manufacturer.js

```
exports.seed = function(knex) {
  // Deletes ALL existing entries
  return knex('table_name').del()
    .then(function () {
      // Inserts seed entries
      return knex('table_name').insert([
        {id: 1, colName: 'rowValue1'},
        {id: 2, colName: 'rowValue2'},
        {id: 3, colName: 'rowValue3'}
      ]);
    });
};
```

Copyright © 2021 TalentLabs Limited

</talentlabs>

Upsert seed data

We want to use upsert instead.
Upsert: Update or Insert.

The upsert operation will

1. Insert a new row if there is no duplication
2. Update the existing row if there is a duplication.

./seeds/initial-manufacturer.js

```
exports.seed = function(knex) {
  return knex.raw(
    `
    insert into manufacturer (id, name)
        values (1, "Lego"), (2, "Disney")
    as new_data
    on duplicate key update
        name=new_data.name;
    `
  );
};
```

Here we define what to update when there is a duplication found.

We want to insert 2 rows,
(1, “Lego”)
(2, “Disney”)

Copyright © 2021 TalentLabs Limited

</talentlabs>

Upsert seed data

We want to use upsert instead.
Upsert: Update or Insert.

The upsert operation will

1. Insert a new row if there is no duplication
2. Update the existing row if there is a duplication.

./seeds/initial-product.js

```
exports.seed = function(knex) {
  return knex.raw(
    `
    insert into product (id, name, price, manufacturer_id)
        values (1, "Product 1", 99.9, 1), (2, "Product 2", 90.2, 2)
    as new_data
    on duplicate key update
        name=new_data.name,
        price=new_data.price,
        manufacturer_id=new_data.manufacturer_id;
    `
  );
};
```

Here we define what to update when there is a duplication found.

We want to insert 2 rows,
(1, "Product 1", 99.9, 1)
(2, "Product 2", 90.2, 2)

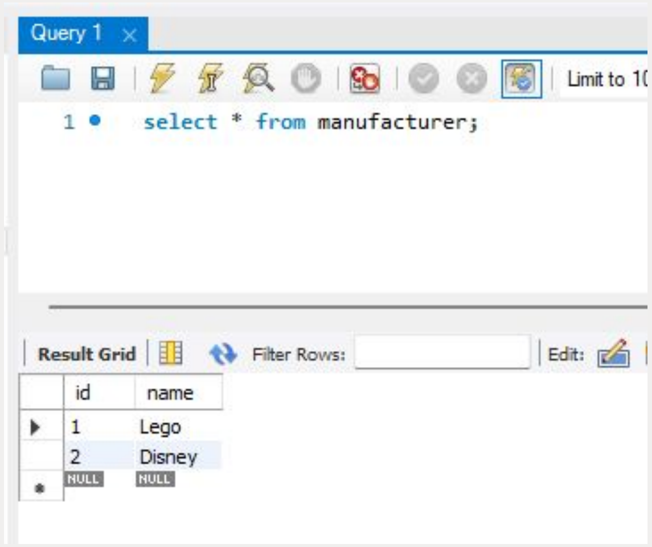
Copyright © 2021 TalentLabs Limited

Execute the seed file

Run these to execute the 2 seed files we created:

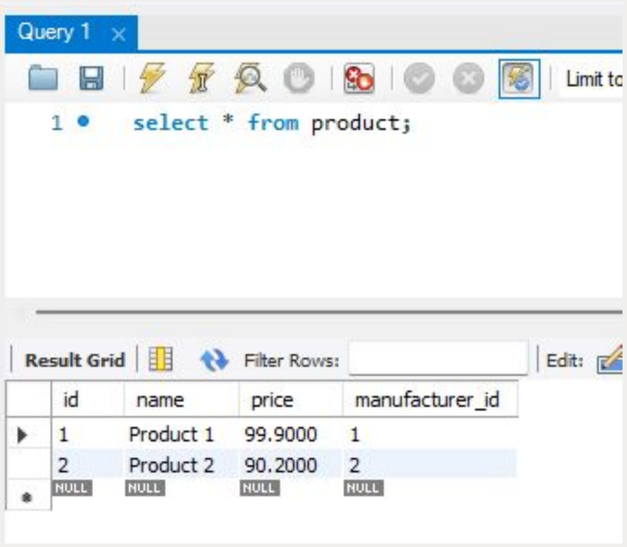
```
npx knex seed:run --specific=initial-manufacturer.js
npx knex seed:run --specific=initial-product.js
```

Check in MySQL Workbench



```
select * from manufacturer;
```

id	name
1	Lego
2	Disney
NULL	NULL



```
select * from product;
```

id	name	price	manufacturer_id
1	Product 1	99.9000	1
2	Product 2	90.2000	2
NULL	NULL	NULL	NULL

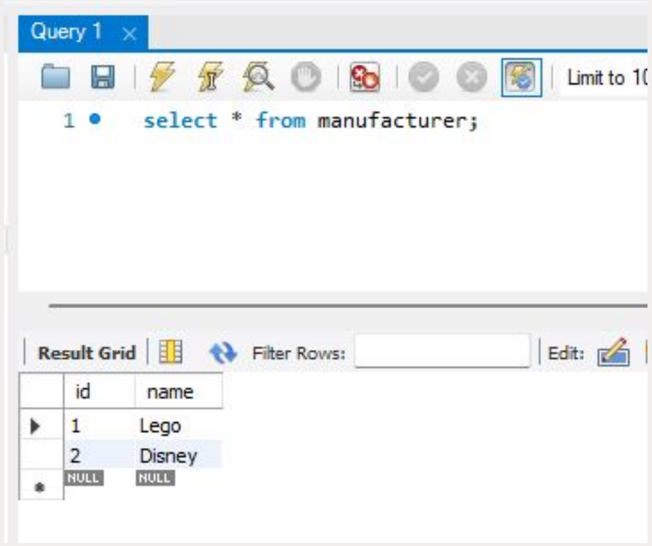
Copyright © 2021 TalentLabs Limited

Test the upsert

Re-run these to execute the 2 seed files we created:

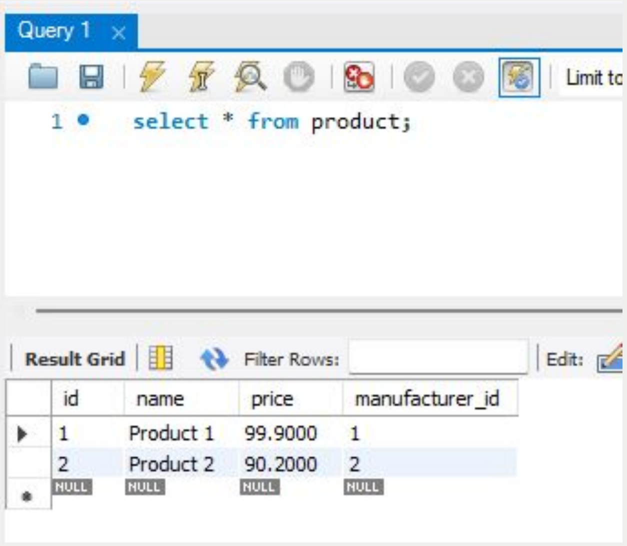
```
npx knex seed:run --specific=initial-manufacturer.js
npx knex seed:run --specific=initial-product.js
```

Check in MySQL Workbench (no new records)



```
select * from manufacturer;
```

id	name
1	Lego
2	Disney
NULL	NULL



```
select * from product;
```

id	name	price	manufacturer_id
1	Product 1	99.9000	1
2	Product 2	90.2000	2
NULL	NULL	NULL	NULL

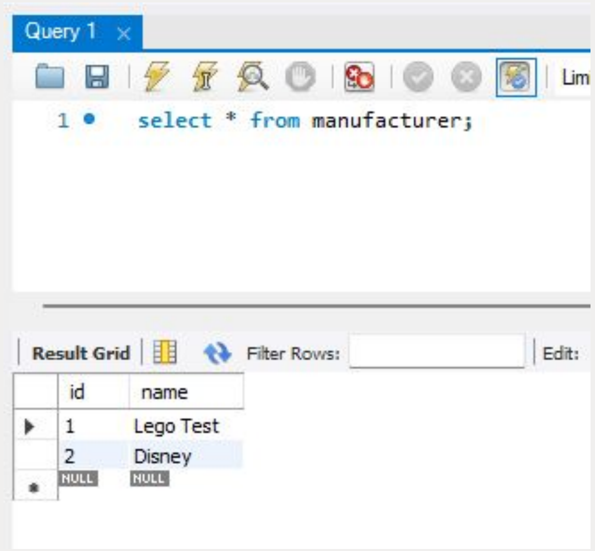
Copyright © 2021 TalentLabs Limited

Test the upsert

If we update the name of the Manufacturer with id = 1 and run the command again.

```
npx knex seed:run --specific=initial-manufacturer.js
```

Check in MySQL Workbench: the name updated (no new records!)



```
select * from manufacturer;
```

id	name
1	Lego Test
2	Disney
NULL	NULL

Copyright © 2021 TalentLabs Limited

Integrate to an Express route

Copyright © 2021 TalentLabs Limited

Initialize a knex connection

Create a file called **./database.js**:

```
const environment = process.env.NODE_ENV || "development";
// Import the knex config from the knexfile.js file.
const config = require("./knexfile");
// Pick the correct database configuration for the environment
// (such as "development")
const environmentConfig = config[environment];
const knex = require("knex");
// Create a Database Connection
const connection = knex(environmentConfig);
module.exports = connection;
```

The knexfile.js

</talentlabs>

Use the knex connection

Import the **./database.js** in the **./routes/index.js**

```
var express = require('express');
var router = express.Router();
var connection = require('../database.js')

/* List manufacturers */
router.get('/manufacturers', function(req, res, next) {
  res.json({
    "manufacturers": [
    ]
  })
});

module.exports = router;
```

Import the knex connection

Return a empty list for now

Use the knex connection

./routes/index.js

```
var express = require('express');
var router = express.Router();
var connection = require('../database.js')

/* List manufacturers */
router.get('/manufacturers', function(req, res, next) {
  //knex connection
  connection
    .raw(`select * from manufacturer;`) // it is a promise
    .then(function (result) {
      var manufacturers = result[0];
      // send back the query result as json
      res.json({
        manufacturers: manufacturers,
      });
    })
    .catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        "message": error
      });
    });
});

module.exports = router;
```

connection.raw: run the SQL statement with the knex database connection. It returns a Promise!

If the SQL statement is executed correctly: we return the SQL results

Otherwise, we notify the client we have an Server error.

Copyright © 2021 TalentLabs Limited

</talentlabs>

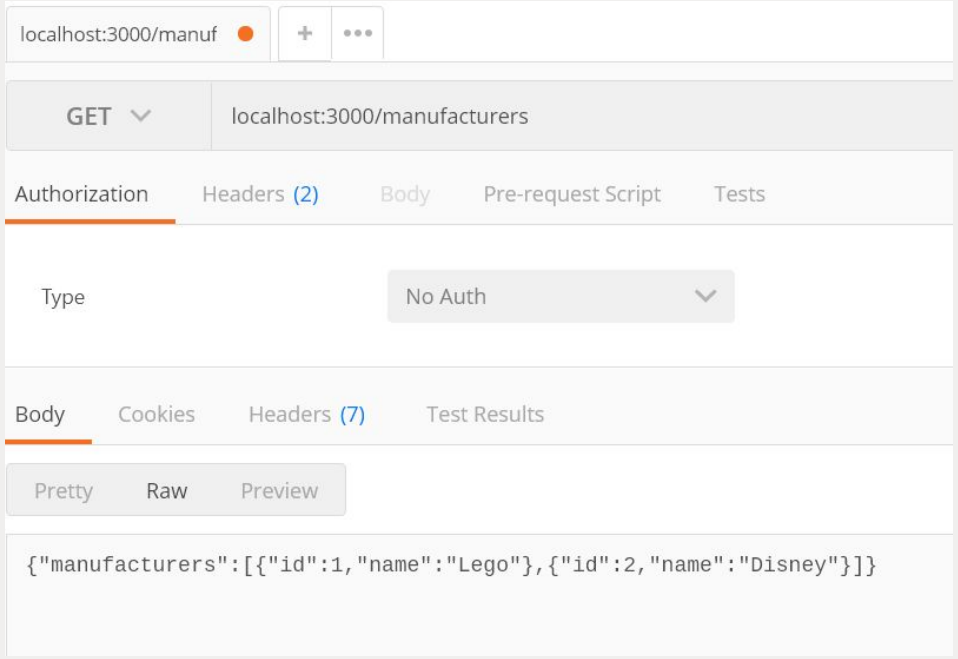
Use the knex connection

./routes/index.js

```
var express = require('express');
var router = express.Router();
var connection = require('../database.js')

/* List manufacturers */
router.get('/manufacturers', function(req, res, next) {
  //knex connection
  connection
    .raw(`select * from manufacturer;`) // it is a promise
    .then(function (result) {
      var manufacturers = result[0];
      // send back the query result as json
      res.json({
        manufacturers: manufacturers,
      });
    })
    .catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        "message": error
      });
    });
});

module.exports = router;
```



localhost:3000/manufacturers

Type: No Auth

```
{"manufacturers":[{"id":1,"name":"Lego"},{"id":2,"name":"Disney"}]}
```

Copyright © 2021 TalentLabs Limited

List, Retrieve, Create, Update, Delete

</talentlabs>

Copyright © 2021 TalentLabs Limited

</talentlabs>

API Endpoints

GET /manufacturers	**List** all manufacturers
GET /manufacturers/:id	**Retrieve** a manufacturer with id = :id
POST /manufacturers	**Create** a new manufacturer
PUT /manufacturers/:id	**Update** a manufacturer with id = :id
DELETE /manufacturers/:id	**Delete** a manufacturer with id = :id

Copyright © 2021 TalentLabs Limited

List manufacturers

./routes/index.js

```
/* List manufacturers */
router.get('/manufacturers', function(req, res, next) {
  //knex connection
  connection
    .raw(`select * from manufacturer;`) // it is a promise
    .then(function (result) {
      var manufacturers = result[0];
      // send back the query result as json
      res.json({
        manufacturers: manufacturers,
      });
    })
    .catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        "message": error
      });
    });
});
```

```
select * from manufacturer
```

connection.raw: run the SQL statement with the knex database connection. It returns a Promise!

If the SQL statement is executed correctly: we return the SQL results

Otherwise, we notify the client we have an Server error.

Copyright © 2021 TalentLabs Limited

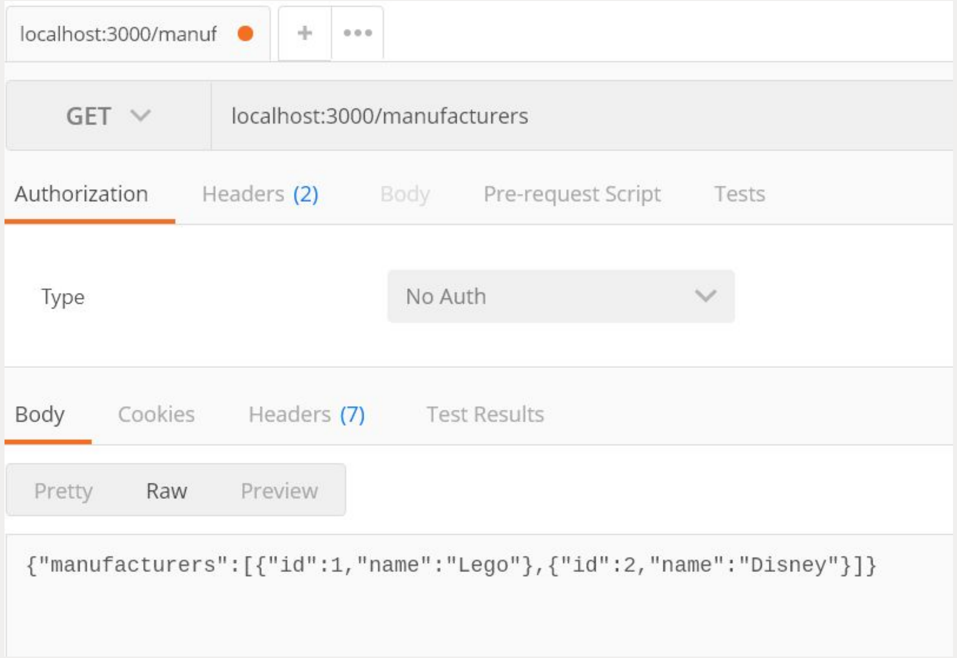
</talentlabs>

List manufacturers

./routes/index.js

```
/* List manufacturers */
router.get('/manufacturers', function(req, res, next) {
  //knex connection
  connection
    .raw(`select * from manufacturer;`) // it is a promise
    .then(function (result) {
      var manufacturers = result[0];
      // send back the query result as json
      res.json({
        manufacturers: manufacturers,
      });
    })
    .catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        "message": error
      });
    });
});
```

```
select * from manufacturer
```



```
{"manufacturers":[{"id":1,"name":"Lego"},{"id":2,"name":"Disney"}]}
```

Copyright © 2021 TalentLabs Limited

</talentlabs>

Retrieve a manufacturer with id = :id

```
select * from manufacturer where id = ?;
```

./routes/index.js

```
router.get('/manufacturers/:id', function(req, res, next) {
  //knex connection
  connection
    .raw(`select * from manufacturer where id = ?`, [req.params["id"]])
    .then(function (result) {
      var manufacturers = result[0];
      // send back the query result as json
      res.json({
        manufacturer: manufacturers[0],
      });
    })
    .catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        "message": error
      });
    });
});
```

Parameter binding:
Req.params["id"] will replace the ?.

connection.raw: run the SQL statement with the knex database connection. It returns a Promise!

If the SQL statement is executed correctly: we return the SQL results

Otherwise, we notify the client we have an Server error.

Copyright © 2021 TalentLabs Limited

</talentlabs>

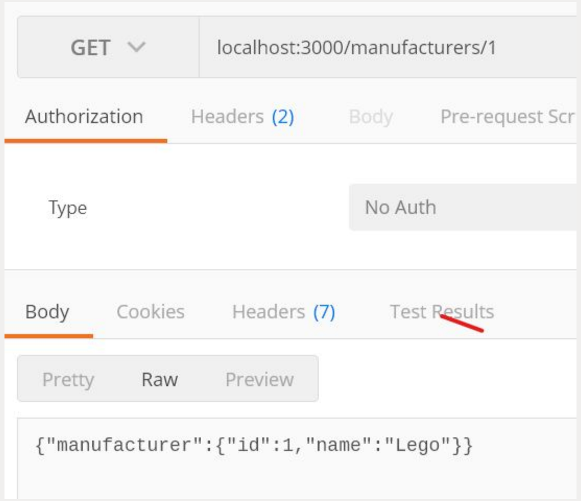
Retrieve a manufacturer with id = :id

```
select * from manufacturer where id = ?;
```

./routes/index.js

```
router.get('/manufacturers/:id', function(req, res, next) {
  //knex connection
  connection
    .raw(`select * from manufacturer where id = ?`, [req.params["id"]])
    .then(function (result) {
      var manufacturers = result[0];
      // send back the query result as json
      res.json({
        manufacturer: manufacturers[0],
      });
    })
    .catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        "message": error
      });
    });
});
```

Parameter binding:
Req.params[“id”] will replace the ?.



```
{"manufacturer":{"id":1,"name":"Lego"}}
```

Copyright © 2021 TalentLabs Limited

Create a new manufacturer

```sql
insert into manufacturer (name) values (?)
```

./routes/index.js

```javascript
router.post('/manufacturers', function(req, res, next) {
    console.log("POST Request", req.body);
    var promise = connection.raw(
      `
      insert into manufacturer (name)
      values (?)
      `,
      [req.body["name"]]
    );
    promise.then(function (result) {
      res.json({
        "message": "Done",
      })
    }).catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        message: error,
      });
    });
});
```

Parameter binding:
req.body[“name”] will replace the ?.

connection.raw: run the SQL statement with the knex database connection. It returns a Promise!

If the SQL statement is executed correctly: we return the SQL results

Otherwise, we notify the client we have an Server error.

Copyright © 2021 TalentLabs Limited

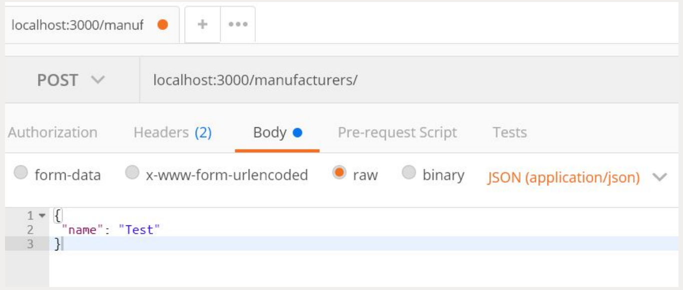
Create a new manufacturer

```sql
insert into manufacturer (name) values (?)
```

Parameter binding:
req.body["name"] will replace the ?.

./routes/index.js

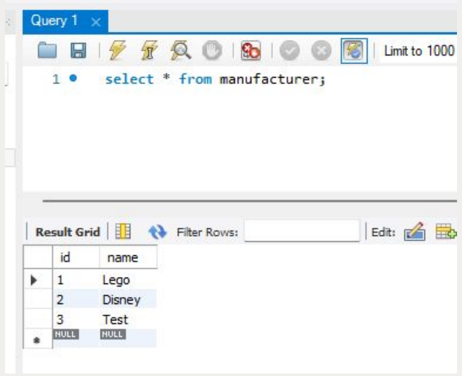
```javascript
router.post('/manufacturers', function(req, res, next) {
   console.log("POST Request", req.body);
   var promise = connection.raw(
     `
     insert into manufacturer (name)
     values (?)
     `,
     [req.body["name"]]
   );
   promise.then(function (result) {
     res.json({
       "message": "Done",
     })
   }).catch(function (error) {
     // log the error
     console.log(error);
     res.json(500, {
       message: error,
     });
   });
});
```



POST localhost:3000/manufacturers/

raw JSON (application/json)

```json
{
  "name": "Test"
}
```



```sql
select * from manufacturer;
```

id	name
1	Lego
2	Disney
3	Test
NULL	NULL

Copyright © 2021 TalentLabs Limited

</talentlabs>

Update a manufacturer with id = :id

```
update into manufacturer set name = ? where id = ?
```

./routes/index.js

```
router.put('/manufacturers/:id', function(req, res, next) {
   console.log("PUT Request", req.body);
   var promise = connection.raw(
     `
     update manufacturer
     set name = ?
     where id = ?
     `,
     [req.body["name"], req.params["id"]]
   );
   promise.then(function (result) {
     res.json({
       "message": "Done",
     })
   }).catch(function (error) {
     // log the error
     console.log(error);
     res.json(500, {
       message: error,
     });
   });
});
```

Parameter binding:
req.body["name"] will replace the first ?
req.params["id"] will replace the second ?

connection.raw: run the SQL statement with the knex database connection. It returns a Promise!

If the SQL statement is executed correctly: we return the SQL results

Otherwise, we notify the client we have an Server error.

Copyright © 2021 TalentLabs Limited

</talentlabs>

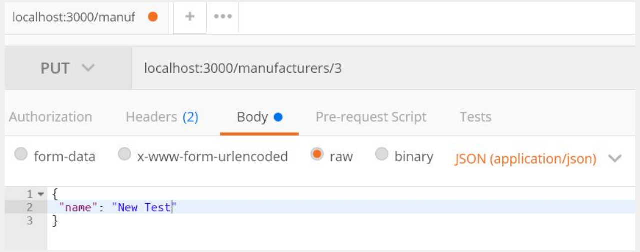
Update a manufacturer with id = :id

```
update into manufacturer set name = ? where id = ?
```

./routes/index.js

```
router.put('/manufacturers/:id', function(req, res, next) {
    console.log("PUT Request", req.body);
    var promise = connection.raw(
      `
        update manufacturer
        set name = ?
        where id = ?
      `,
        [req.body["name"], req.params["id"]]
    );
    promise.then(function (result) {
      res.json({
        "message": "Done",
      })
    }).catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        message: error,
      });
    });
});
```

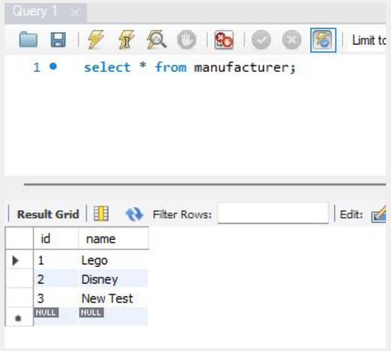
Parameter binding:
req.body[“name”] will replace the first ?
req.params[“id”] will replace the second ?



PUT localhost:3000/manufacturers/3

Body: raw JSON (application/json)

```
{
  "name": "New Test"
}
```



```
select * from manufacturer;
```

id	name
1	Lego
2	Disney
3	New Test
NULL	NULL

</talentlabs>

Delete a manufacturer with id = :id

```
delete from manufacturer set where id = ?
```

./routes/index.js

```
router.delete('/manufacturers/:id', function(req, res, next) {
   var promise = connection.raw(
     `
     delete from manufacturer
     where id = ?
     `,
     [req.params["id"]]
   );
   promise.then(function (result) {
     res.json({
       "message": "Done",
     })
   }).catch(function (error) {
     // log the error
     console.log(error);
     res.json(500, {
       message: error,
     });
   });
});
```

Parameter binding:
req.params[“id”] will replace the ?

connection.raw: run the SQL statement with the knex database connection. It returns a Promise!

If the SQL statement is executed correctly: we return the SQL results

Otherwise, we notify the client we have an Server error.

Copyright © 2021 TalentLabs Limited

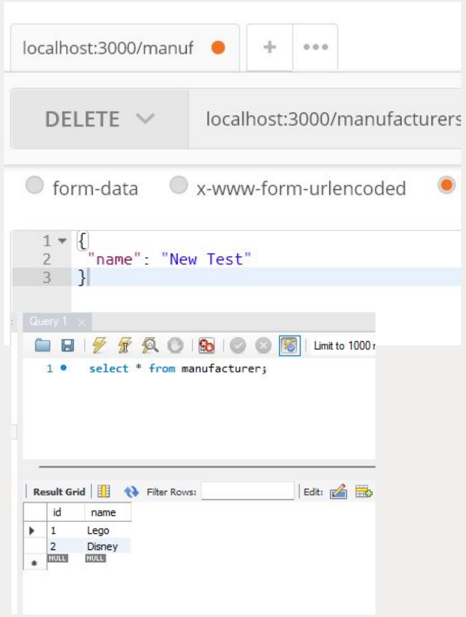
Delete a manufacturer with id = :id

./routes/index.js

```
router.delete('/manufacturers/:id', function(req, res, next) {
    var promise = connection.raw(
      `
      delete from manufacturer
      where id = ?
      `,
      [req.params["id"]]
    );
    promise.then(function (result) {
      res.json({
        "message": "Done",
      })
    }).catch(function (error) {
      // log the error
      console.log(error);
      res.json(500, {
        message: error,
      });
    });
});
```

```
delete from manufacturer set where id = ?
```

Parameter binding:
req.params["id"] will replace the ?



localhost:3000/manufacturers

```
{
  "name": "New Test"
}
```

```
select * from manufacturer;
```

id	name
1	Lego
2	Disney
NULL	NULL

Copyright © 2021 TalentLabs Limited