NODEJS

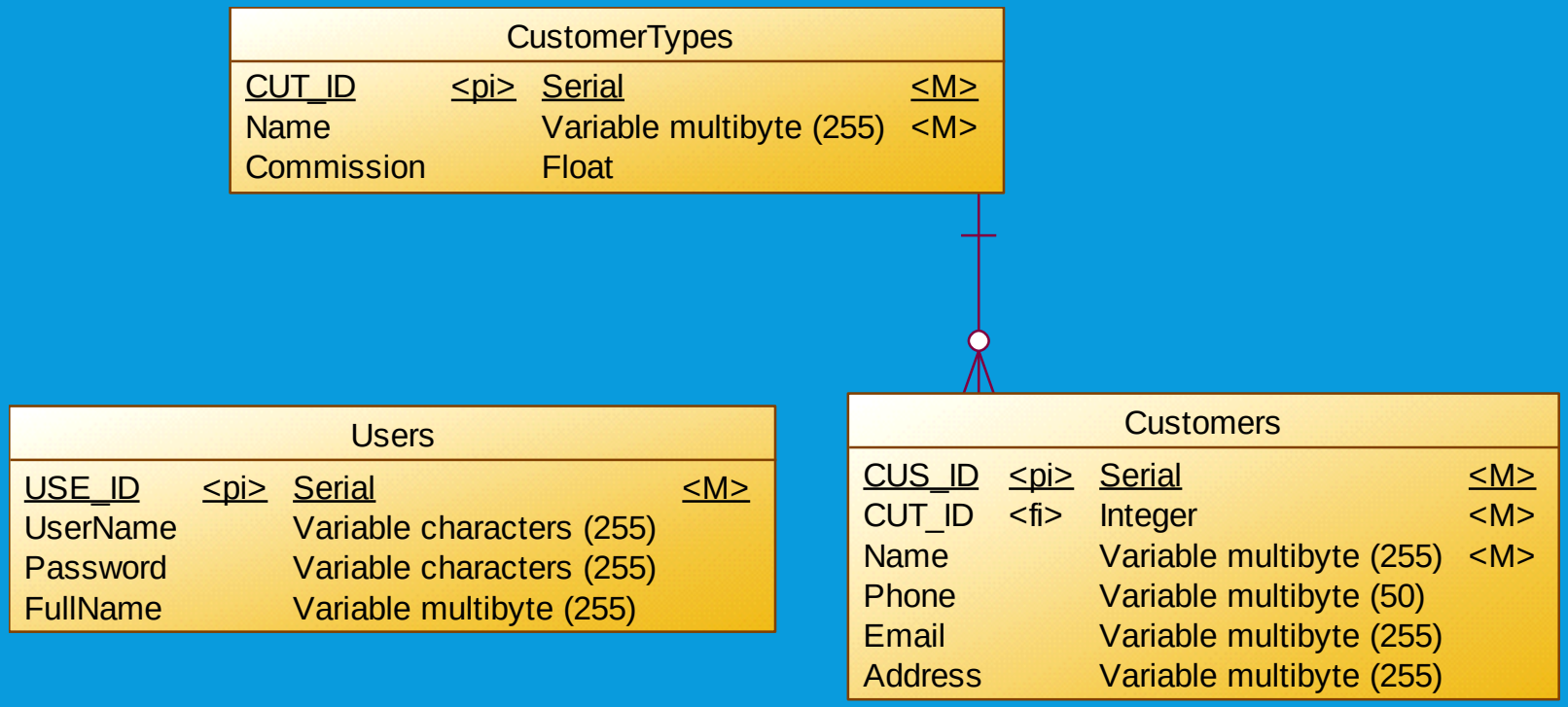
Express Framework

Building simple web service with MSSQL

OBJECTIVES

1. Preparing Database
2. Installing new packages
3. Creating model classes
4. Creating controller classes
5. Optimizing api result

1. PREPARING DATABASE



2. INSTALLING PACKAGES

- In command prompt, type:

  npm install --save sequelize tedious

- Create new folders named "/models", "/controllers"

3. CREATING MODEL CLASSES

- Create new files in "models" folder
  - "db.js"
  - "customer_type.js"
  - "customer.js"
  - "user.js"

IN USER.JS FILE

```
module.exports = (sequelize, type) => {
    return sequelize.define('Users', {
        id: {
            field: 'USE_ID',
            type: type.INTEGER,
            primaryKey: true,
            autoIncrement: true
        },
        userName: type.STRING,
        password: type.STRING,
        fullName: type.STRING
    }, { timestamps: false })
}
```

IN CUSTOMER_TYPE.JS FILE

```
module.exports = (sequelize, type) => {
    return sequelize.define('CustomerTypes', {
        id: {
            field: 'CUT_ID',
            type: type.INTEGER,
            primaryKey: true,
            autoIncrement: true
        },
        name: {
            type: type.STRING,
            allowNull: false
        },
        commission: {
            type: type.FLOAT,
            defaultValue: 0.0
        },
    }, {timestamps: false})
}
```

IN CUSTOMER.JS FILE

```
module.exports = (sequelize, type) => {
    return sequelize.define('Customers', {
        id: {
            field: 'CUS_ID',
            type: type.INTEGER,
            primaryKey: true,
            autoIncrement: true
        },
        CUT_ID: {  type: type.INTEGER,  allowNull: false },
        name: { type: type.STRING, allowNull: false },
        phone: type.STRING(50),
        email: type.STRING,
        address: type.STRING,
    }, { timestamps: false })
}
```

IN DB.JS FILE

```
const Sequelize = require('sequelize');
const UserModel = require('./user')
const CustomerTypeModel = require('./customer_type')
const CustomerModel = require('./customer')

const sequelize = new Sequelize('DBName', 'username', 'password', {
    dialect: 'mssql',
    host: 'localhost',
    dialectOptions: {
      options: {
          instanceName: 'SQLEXPRESS'
      }
    },
    pool: {  max: 20,  min: 0,  acquire: 30000,  idle: 10000  },
    logging: true
});
```

IN DB.JS FILE (CONT.)

```
const User = UserModel(sequelize, Sequelize)
const CustomerType = CustomerTypeModel(sequelize, Sequelize)
const Customer = CustomerModel(sequelize, Sequelize)

Customer.belongsTo(CustomerType, {foreignKey: 'CUT_ID', as: 'customerType'});
CustomerType.hasMany(Customer, {foreignKey: 'CUT_ID', as: 'customers'});

// only run once, then comment out
sequelize.sync({ force: true }).then(() => {
    console.log(`Database & tables created!`)
});

module.exports = {
  User,
  CustomerType,
  Customer
}
```

4. CREATING CONTROLLER CLASSES

4.1. Customer type controller

4.2. Customer controller

4.3. User controller

4.1. CUSTOMER TYPE CONTROLLER

4.1.1. Creating "customer_types" controller

4.1.2. Listing all customer types (GET)

4.1.3. Getting a specific customer type (GET)

4.1.4. Creating a new customer type (POST)

4.1.5. Updating an existing customer type (PUT)

4.1.6. Deleting a customer type (DELETE)

4.1.1. CREATING "CUSTOMER_TYPES" CONTROLLER

- Create a new file named "customer_types.js" in "constrollers" folder
- Type code blocks below

```
const express = require('express');
const { CustomerType } = require('./../models/db')
const router = express.Router();
router.use((req, res, next) => {
    // authorize here
    next();
});

// fill customer type apis here

module.exports = router;
```

4.1.2. LISTING ALL CUSTOMER TYPES

```
const express = require('express');
const { CustomerType } = require('./../models/db')
const router = express.Router();
router.use((req, res, next) => {
    // authorize here
    next();
});

// fill customer apis here
router.get('/', (req, res) => {
    CustomerType.findAll().then(types => {
        res.json(types)
    });
});

module.exports = router;
```

4.1.3. GETTING A SPECIFIC CUSTOMER TYPE

```
router.get('/', (req, res) => {
    CustomerType.findAll().then(types => {
        res.json(types)
    });
});

router.get('/:id', (req, res) => {
    CustomerType.findByPk(req.params.id).then(type => {
        if (type != null) {
            res.json(type);
        } else {
            res.status(404).send('Not Found!');
        }
    });
});

module.exports = router;
```

4.1.4. CREATING A NEW CUSTOMER TYPE

```
> router.get('/', (req, res) => {···
  });

> router.get('/:id', (req, res) => {···
  });

  router.post('/', (req, res) => {
      //validate data here
      CustomerType.create(req.body).then(type => {
          res.json(type);
      }).catch(err => {
          return res.status(400).send(err.errors);
      });
  });

  module.exports = router;
```

4.1.5. UPDATING AN EXISTING CUSTOMER TYPE

```
> router.post('/', (req, res) => {···
  });
```

```
router.put('/:id', (req, res) => {
    //validate data here
    CustomerType.findByPk(req.params.id).then(type => {
        if (type != null) {
            type.update({
                name: req.body.name,
                commission: req.body.commission
            }).then(type => {
                res.json(type);
            }).catch(err => {
                return res.status(400).send(err.errors);
            });
        } else {
            res.status(404).send('Not Found!');
        }
    });
});
```

4.1.6. DELETING A CUSTOMER TYPE

```
> router.put('/:id', (req, res) => {···
  });

  router.delete('/:id', (req, res) => {
      CustomerType.destroy({
          where: {
              id: req.params.id
          }
      }).then(type => {
          res.json(type);
      }).catch(err => {
          return res.status(500).send(err.errors);
      });
  });

  module.exports = router;
```

IN INDEX.JS FILE

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use((req, res, next) => {
    res.header('Access-Control-Allow-Origin','*');
    res.header('Access-Control-Allow-Headers','*');
    res.header('Access-Control-Allow-Methods','*');
    next();
});
app.use('/img', express.static(__dirname+'/data'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));

const customerTypeCtrl = require('./controllers/customer_types');
app.use('/customerTypes', customerTypeCtrl);

// invalid Url
app.use((req, res) => {
    res.status(404).send('Not Found!');
});

const server = app.listen(8081, () => {
    const host = server.address().address;
    const port = server.address().port;
    console.log('Server is running at http://%s:%s', host, port);
});
```

TEST CUSTOMER TYPE APIS

- Use POSTMAN to test customer type apis

4.2. CUSTOMER CONTROLLER

4.2.1. Creating "customers" controller

4.2.2. Listing all customers (GET)

4.2.3. Getting a specific customer (GET)

4.2.4. Creating a new customer (POST)

4.2.5. Updating an existing customer (PUT)

4.2.6. Deleting a customer (DELETE)

4.2.1. CREATING "CUSTOMERS" CONTROLLER

- Create a new file named "customers.js" in "controllers" folder
- Type code blocks below

```
const express = require('express');
const { CustomerType, Customer } = require('./../models/db')
const router = express.Router();
router.use((req, res, next) => {
    // authorize here
    next();
});

// fill customer apis here

module.exports = router;
```

4.2.2. LISTING ALL CUSTOMERS (GET)

```
const express = require('express');
const { CustomerType, Customer } = require('./../models/db')
const router = express.Router();
router.use((req, res, next) => {
    // authorize here
    next();
});

// fill customer apis here
router.get('/', (req, res) => {
    Customer.findAll({
        include: [{
            model: CustomerType,
            as: 'customerType'
        }]
    }).then(types => {
        res.json(types)
    });
});

module.exports = router;
```

4.2.3. Getting a specific customer (GET)

4.2.4. Creating a new customer (POST)

4.2.5. Updating an existing customer (PUT)

4.2.6. Deleting a customer (DELETE)

EXERCISE

4.3. USER CONTROLLER

4.3.1. Creating "helper" module

4.3.2. Creating "users" controller

4.3.3. Creating a new user (POST)

4.3.4. Creating login api (POST)

4.3.1. CREATING "HELPER" MODULE

- Create a new folder named "/utils"
- In "/utils" folder, create a new file named "helper.js", then type below code

```
const crypto = require('crypto');

const createHash = (text) => {
    return crypto.createHash('sha256')
                 .update(text, 'utf8').digest('base64');
};

module.exports = {
    hash: createHash
}
```

4.3.1. CREATING "USERS" CONTROLLER

- Create a new file named "users.js" in "controllers" folder
- Type code blocks below

```
const express = require('express');
const crypt = require('../utils/helper');
const { User } = require('./../models/db');
const router = express.Router();
router.use((req, res, next) => {
    // authorize here
    next();
});

// fill user apis here

module.exports = router;
```

4.3.3. CREATING A NEW USER (POST)

```
const express = require('express');
const crypt = require('../utils/helper');
const { User } = require('./../models/db');
const router = express.Router();
router.use((req, res, next) => {···
});

// fill user apis here
router.post('/', (req, res) => {
    //validate data here
    req.body.password = crypt.hash(req.body.password);
    User.create(req.body).then(type => {
        res.json(type);
    }).catch(err => {
        return res.status(400).send(err.errors);
    });
});

module.exports = router;
```

4.3.4. CREATING LOGIN API (POST)

```
// fill user apis here
router.post('/', (req, res) => {···
});

router.post('/login', (req, res) => {
    User.findOne({
        where: {
            userName: req.body.userName,
            password: crypt.hash(req.body.password)
        }
    }).then(aUser => {
        if (aUser != null) {
            res.json({
                id: aUser.id,
                userName: aUser.userName,
                fullName: aUser.fullName
            });
        } else {
            res.status(401).send('Invalid username or password');
        }
    });
});
```

TEST USER APIS

5. OPTIMIZING API RESULT

- 5.1. Creating "response_result" module
- 5.2. Modifying api methods

4.1. CREATING RESPONSE-RESULT MODULE

- Create a new file named "base_response.js" in "utils" folder then type below code

```
const result = (json) => ({ errorCode: 0, data: json });
const error = (code, mess) => ({
    errorCode: code,
    message: mess
});

module.exports = {
    Result: result,
    ErrorResult: error
}
```

4.2. MODIFYING API METHODS

```
const express = require('express');
const { CustomerType } = require('./../models/db');
const {ErrorResult, Result} = require('./../utils/base_response')
const router = express.Router();
router.use((req, res, next) => {···
});

router.get('/', (req, res) => {
    CustomerType.findAll().then(types => {
        res.json(Result(types))
    });
});

router.get('/:id(\\d+)', (req, res) => {
    CustomerType.findByPk(req.params.id).then(type => {
        if (type != null) {
            res.json(Result(type));
        } else {
            res.status(404).json(ErrorResult(404, 'Not Found!'));
        }
    });
});
```

```
router.post('/', (req, res) => {
    //validate data here
    CustomerType.create(req.body).then(type => {
        res.json(Result(type));
    }).catch(err => {
        return res.status(400).send(ErrorResult(400, err.errors));
    });
});

router.put('/:id(\\d+)', (req, res) => {
    //validate data here
    CustomerType.findByPk(req.params.id).then(type => {
        if (type != null) {
            type.update({
                name: req.body.name,
                commission: req.body.commission
            }).then(type => {
                res.json(Result(type));
            }).catch(err => {
                return res.status(400).send(ErrorResult(400, err.errors));
            });
        } else {
            res.status(404).send(ErrorResult(404, 'Not Found!'));
        }
    });
});
```

```
router.delete('/:id(\\d+)', (req, res) => {
    CustomerType.destroy({
        where: {
            id: req.params.id
        }
    }).then(type => {
        res.json(Result(type));
    }).catch(err => {
        return res.status(500).send(ErrorResult(500, err.errors));
    });
});
```

Update Result & ErrorResult to other return results.