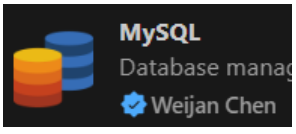
# Sequelize-join (배포)

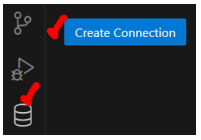


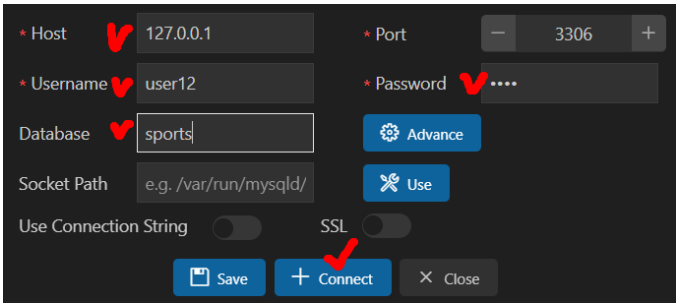
# 1. 작업설정

## 1-1 실행순서

- npm install  npm install express ejs nodemon mysql2 sequelize  dotenv
- cmd → mysql -uroot -p  → 새로운 유저 생성 (user12) → “sports” 디비생성 → 테이블생성 및 데이터 입력
- 확장프로그램 설치

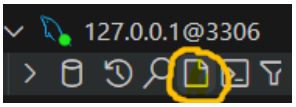



- ERD (entity-relationship diagram) 내용에 따라 테이블을 작성한다.

- 배포된 쿼리문을 실행하여 테이블과 데이터를 입력한다
- 새로운 쿼리문 생성하여 테이블의 내용을 확인한다 (




```
select * from player;
select * from profile;
select * from team;
select * from game;
select * from teamgame;
```

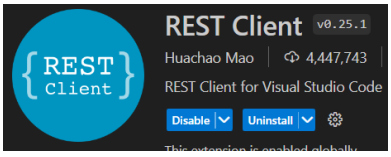
- 디비 연결을 종료한다.

## 1- 2. .env

- process.env 파일 참조

## 1-3. HTTP API란?

HTTP를 사용하여 프로그램끼리 소통하는 API를 말한다. 보통 우리가 흔히 보는 OPEN API, facebook API, kakao API 등의 대부분 API는 HTTP라는 통신 규칙으로 소통하는 API이다.




REST Client 는 `.http` 또는 `.rest` 를 확장자로 갖는 파일에서 HTTP 요청을 작성하고 파일 내부에서 요청을 실행할 수 있다.

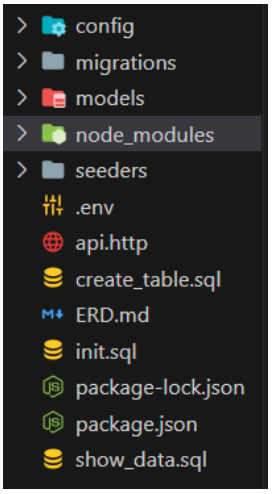
```
[api.http]
Send Request //아래의코드를 붙여넣기해서 왼쪽글자가 나타나면성공
GET  https://jsonplaceholder.typicode.com/todos/

### 이제 Send Request 클릭하며 REST Client가
     VS CODE 내부에서 직접 URL에 쵸청하여 응답결과를 보여준다
### 이제 post 요청을 해보자
```

```
Send Request
POST https://jsonplaceholder.typicode.com/posts HTTP/1.1
Content-Type: application/json; charset=UTF-8

{
  "name": "sample",
  "time": "Wed, 21 Oct 2015 18:27:50 GMT"
}

### 실행결과는 출력화면 하단을 봐야한다
```

아래와 같이 이미 작성한 GET 요청 아래에 `###` 을 추가하자. `###` 로 구분하지 않는다면, REST Client 가 두개의 요청을 구분하지 못한다.

# 2. sequelize 기초파일완성

```
> npm install sequelize
> npx sequelize init
```




실습파일 📁 시퀄(413)-1

실습파일 📁 시퀄(413) -join → 확장프로그램 mysql 열고 insertData.sql 실행

http:127.0.0.1:8000으로 첫화면 조회후  api.http 파일로 조회를 합니다

# 3. sequelize 를 이용한 관계설정

모델 정의는 sequelize 내장 함수 `define()` 을 사용한다

살펴볼 관계를 맺는 메소드는 belongsTo(), hasOne(), hasMany()

## 3-1. 1:1 관계: 하나의 외래 키로 연결된 두 모델 간의 연결 관계

[User] 모델

```
module.exports = (sequelize, DataTypes) => {

    const Users = sequelize.define("Users", {
          id: {       },
          email: {    },
          name: {    },
          phone: {     },
    }, {
      charset: "utf8", // 한국어 설정
      collate: "utf8_general_ci", // 한국어 설정
      tableName: "Users", // 테이블 이름
      timestamps: true, // createAt & updateAt 활성화
      paranoid: true, // timestamps 가 활성화 되어야 사용 가능 > d
    });

    Users.associate = models => {
        /**
         * Users안에 있는 "id값"을 "user_id라는 컬럼 이름"으로
         * UserInfo모델에 새로운 컬럼으로 추가한다.
         */
        Users.hasOne(models.UserInfo, {foreignKey: "user_id",
    };                                                          sourceKey: "id"});
    return Users;
};
```

```
module.exports = (sequelize, DataTypes) => {

    const UserInfo = sequelize.define("UserInfo", {
      id: { },
      permission: {},
      status: {  },
      block: {   },
      account_expired: {   },
      password_expired: {  },
    }, {
```

```
      charset: "utf8", // 한국어 설정
      collate: "utf8_general_ci", // 한국어 설정
      tableName: "UserInfo", // 테이블 이름
      timestamps: true, // createAt & updateAt 활성화
      paranoid: true, // timestamps 가 활성화 되어야 사용 가능 > d
    });

    UserInfo.associate = models => {
        /**
         * UserInfo모델 안에 "user_id라는 컬럼 이름"으로 Users모델에
         * "id값"을 새로운 컬럼으로 추가한다.
         */
        UserInfo.belongsTo(models.Users, "user_id", sourceKey: "id"});
    };

    return UserInfo;
};
```

**hasOne** : 관계를 맺는 대상(자식)에게 자신의 외래 키를 추가합니다. (Users ⇒ UserInfo) UserInfo 외래 키가 추가됩니다.

**belongsTo** : 관계를 맺는 대상(부모)에게 외래 키를 받아 추가합니다. (UserInfo ⇒ Users) UserInfo 외래 키가 추가됩니다.

```
mysql> desc Users;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | char(36)     | NO   | PRI | NULL    |       |
| email     | varchar(100) | YES  |     | NULL    |       |
| password  | varchar(60)  | YES  |     | NULL    |       |
| name      | varchar(100) | YES  |     | NULL    |       |
| phone     | varchar(72)  | YES  |     | NULL    |       |
| createdAt | datetime     | NO   |     | NULL    |       |
| updatedAt | datetime     | NO   |     | NULL    |       |
| deletedAt | datetime     | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Users 모델

```
mysql> desc UserInfo;
+------------------+-----------------------+------+-----+---------+-------+
| Field            | Type                  | Null | Key | Default | Extra |
+------------------+-----------------------+------+-----+---------+-------+
| id               | char(36)              | NO   | PRI | NULL    |       |
| permission       | varchar(45)           | YES  |     | NULL    |       |
| status           | enum('PAUSE','ACTIVE') | NO   |     | PAUSE   |       |
| block            | tinyint(1)            | YES  |     | 0       |       |
| account_expired  | varchar(45)           | YES  |     | NULL    |       |
| password_expired | varchar(45)           | YES  |     | NULL    |       |
| createdAt        | datetime              | NO   |     | NULL    |       |
| updatedAt        | datetime              | NO   |     | NULL    |       |
| deletedAt        | datetime              | YES  |     | NULL    |       |
| user_id          | char(36)              | YES  | MUL | NULL    |       |
+------------------+-----------------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

💊 Userinfo 는 Users에게 속했기  때문에 (belongsTo) sourceKey : (부모의) id 가 되고 , 나 UserInfo 에게 “user_id” 라는 이름의 외래키로 추가한다

## 3-2. 1:m 관계

**hasMany** : 관계를 맞는 대상에게 자신의 외래키를 추가한다. 그리고 복수개의 데이터를 추가하기가 가능합니다

**belongsTo** : 관계를 맺는 대상에게 외래키를 받아 추가합니다.

```
[CompanyInfomation 모델]
module.exports = (sequelize, DataTypes) => {
  const CompanyInformation = sequelize.define("CompanyInforma
```

```
        id: { },
        hospital_name:{ },
        email: {   },
         name: {   },
        phone: {   },
        fax: {   },
        business_number: {  },
        address: {    },
        package: {   },
   }, {
    charset: "utf8", // 한국어 설정
    collate: "utf8_general_ci", // 한국어 설정
    tableName: "CompanyInformation", // 테이블 이름
    timestamps: true, // createAt & updateAt 활성화
  });

  CompanyInformation.associate = models => {

    /**
     * CompanyInformation안에 있는 "id값"을 "company_id라는 컬럼 C
     * Users모델에 새로운 컬럼으로 추가한다.
     */
CompanyInformation.hasMany(models.Users, {foreignKey : "compa
  };

  return CompanyInformation;
};
```

```
[Users모델]
module.exports = (sequelize, DataTypes) => {
    const Users = sequelize.define("Users", {
      id: {  },
      email: { },
      password: {  },
      name: { },
      phone: {   },
```

```
    }, {
      charset: "utf8", // 한국어 설정
      collate: "utf8_general_ci", // 한국어 설정
      tableName: "Users", // 테이블 이름
      timestamps: true, // createAt & updateAt 활성화
      paranoid: true, // timestamps 가 활성화 되어야 사용 가능 > d
    });

  Users.associate = models => {
        /**
         * Users안에 있는 "id값"을 "user_id라는 컬럼 이름"으로
         * UserInfo모델에 새로운 컬럼으로 추가한다.
         */
    Users.hasOne(models.UserInfo, {foreignKey: "user_id", sou


        /**
         *  Users모델 안에 "company_id라는 컬럼 이름"으로
         *  CompanyInformation모델에 있는 "id값"을 새로운 컬럼으로
         */
    Users.belongsTo(models.CompanyInformation,{foreignKey:"co
    };

    return Users;
};
```

```
mysql> desc CompanyInformation;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| id              | char(36)     | NO   | PRI | NULL    |       |
| hospital_name   | varchar(255) | YES  |     | NULL    |       |
| email           | varchar(100) | YES  |     | NULL    |       |
| name            | varchar(100) | YES  |     | NULL    |       |
| phone           | varchar(72)  | YES  |     | NULL    |       |
| fax             | varchar(72)  | YES  |     | NULL    |       |
| business_number | varchar(45)  | YES  |     | NULL    |       |
| address         | text         | YES  |     | NULL    |       |
| package         | json         | YES  |     | NULL    |       |
| createdAt       | datetime     | NO   |     | NULL    |       |
| updatedAt       | datetime     | NO   |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
11 rows in set (0.00 sec)
```

CompanyInformation 모델

```
mysql> desc Users;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| id         | char(36)     | NO   | PRI | NULL    |       |
| email      | varchar(100) | YES  |     | NULL    |       |
| password   | varchar(60)  | YES  |     | NULL    |       |
| name       | varchar(100) | YES  |     | NULL    |       |
| phone      | varchar(72)  | YES  |     | NULL    |       |
| createdAt  | datetime     | NO   |     | NULL    |       |
| updatedAt  | datetime     | NO   |     | NULL    |       |
| deletedAt  | datetime     | YES  |     | NULL    |       |
| company_id | char(36)     | YES  | MUL | NULL    |       |
+------------+--------------+------+-----+---------+-------+
9 rows in set (0.01 sec)
```

Users 모델

- **위의 관계를 바탕으로 category 모델을 생성**

```
[category 모델]
module.exports = (sequelize, DataTypes) => {

    const Category = sequelize.define("Category", {
      id: { },
      name: { },
      }, {   });
```

```
      Category.associate = models => {
       Category.belongsToMany(models.Product, {through: "Prod
    };

    return Category;
};
```

```
[product 모델]
module.exports = (sequelize, DataTypes) => {

    const Product = sequelize.define("Product", {
      id: { },
      name: {    },
    }, });

    Product.associate = models => {
     Product.belongsToMany(models.Category, {through: "Produc
    };

    return Product;
};
```

```
mysql> show tables;
+--------------------+
| Tables_in_ryan     |
+--------------------+
| Category           |
| CompanyInformation |
| Product            |
| ProductCategory    |
| UserInfo           |
| Users              |
+--------------------+
6 rows in set (0.00 sec)
```

```
mysql> desc ProductCategory;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| createdAt  | datetime | NO   |     | NULL    |       |
| updatedAt  | datetime | NO   |     | NULL    |       |
| CategoryId | char(36) | NO   | PRI | NULL    |       |
| ProductId  | char(36) | NO   | PRI | NULL    |       |
+------------+----------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

ProductCategory 테이블이 존재하며 칼럼으로 각각에 primary key 가 생성되어 있다.

- **직접 관계를 설정해준다**

```
[Category 모델]
module.exports = (sequelize, DataTypes) => {

    const Category = sequelize.define("Category", {
      id: {     },
      name: {    },
    }, { });

Category.associate = models => {
Category.belongsToMany(models.Product, {
     through: "ProductCategory", foreignKey: "category_id", s
    };
    return Category;
};
```

```
  [Product 모델]
  module.exports = (sequelize, DataTypes) => {

    const Product = sequelize.define("Product", {
      id: {  },
      name: { },
    }, { });

  Product.associate = models => {
     Product.belongsToMany(models.Category, {
     through: "ProductCategory", foreignKey: "product_id", so
    };

    return Product;
```

```
[ProductCategory 모델]
module.exports = (sequelize, DataTypes) => {

    const ProductCategory = sequelize.define("ProductCategory
```

```
      id: {       },
      name: {       },
    }, { });

    ProductCategory.associate = models => {
      ProductCategory.belongsTo(models.Product, {
        foreignKey: 'product_id', sourceKey: "id"
      });
      ProductCategory.belongsTo(models.Category, {
        foreignKey: 'category_id', sourceKey: "id"
      });
    };

    return ProductCategory;
};
```

```
mysql> show tables;
+--------------------+
| Tables_in_ryan     |
+--------------------+
| Category           |
| CompanyInformation |
| Product            |
| ProductCategory    |
| UserInfo           |
| Users              |
+--------------------+
6 rows in set (0.00 sec)
```

```
mysql> desc ProductCategory;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| id          | char(36)     | NO   | PRI | NULL    |       |
| name        | varchar(100) | YES  |     | NULL    |       |
| createdAt   | datetime     | NO   |     | NULL    |       |
| updatedAt   | datetime     | NO   |     | NULL    |       |
| deletedAt   | datetime     | YES  |     | NULL    |       |
| category_id | char(36)     | YES  | MUL | NULL    |       |
| product_id  | char(36)     | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

동일한 테이블이 생성되지만, ProductCategory테이블의 정보는 다르다.

참조 :

https://hudi.blog/vscode-rest-client/

https://blog.jiniworld.me/144#google_vignette

https://any-ting.tistory.com/51