1. 템플릿 만들기
GET /admin/shops/detail/:id

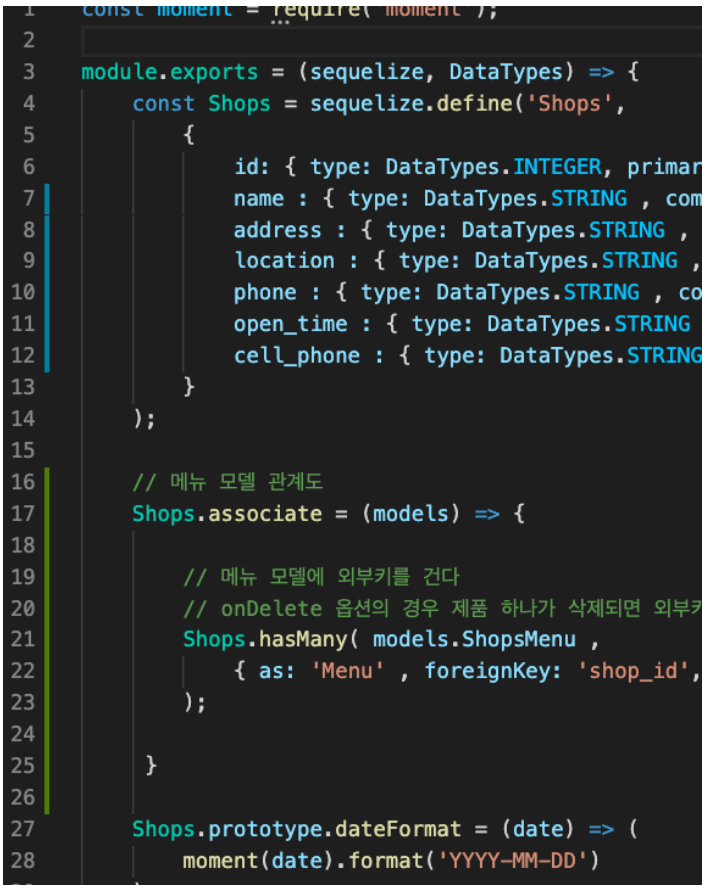
아래와 같은 화면을 만들기

tempate/admin/detail.html

```
<hr />

        메뉴 작성하기

      <form action="" method="post">
        <div class="col-xs-3">
          메뉴명 : <input type="text" class="form-control" name="name" required />
        </div>

        <div class="col-xs-3">
          가격 : <input type="text" class="form-control" name="price" required />
        </div>
        <div class="col-xs-3">
          <button class="btn btn-primary" style="margin-top: 19px">메뉴작성</button>
        </div>
      </form>
```

2. 모델작성
models/Shops.js

```
// 메뉴 모델 관계도
  Shops.associate = (models) => {

    // 메뉴 모델에 외부키를 건다
    // onDelete 옵션의 경우 제품 하나가 삭제되면 외부키가 걸린 메모들도 싹다
삭제해준다
    Shops.hasMany( models.ShopsMenu ,
      { as: 'Menu' , foreignKey: 'shop_id', sourceKey: 'id' , onDelete: 'CASCADE' }
    );

  }
```



models/ShopsMenu.js

```
module.exports = (sequelize, DataTypes) => {
 const ShopsMenu = sequelize.define('ShopsMenu',
    {
        id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
        name : { type: DataTypes.STRING , comment: '메뉴명' },
        price : { type: DataTypes.INTEGER , comment: '가격' }
    },{
        tableName: 'ShopsMenu'
    }
 );

 return ShopsMenu;
}
```

app.js DB sync -> 주석해제 -> 저장 -> 다시 주석

```
38
39
40      }
41
42      dbConnection(){
43          // DB authentication
44          db.sequelize.authenticate()
45          .then(() => {
46              console.log('Connection has been estab
47              return db.sequelize.sync();
48          })
49          .then(() => {
50              console.log('DB Sync complete.');
51          })
52          .catch(err => {
53              console.error('Unable to connect to th
54          });
55      }
```

ShopsMenu 확인

```
[nodemon] starting `node server.js`
Express listening on port 3000
Executing (default): SELECT 1+1 AS result
Connection has been established successfully.
Executing (default): CREATE TABLE IF NOT EXISTS `Shops` (`id` INTEGER auto
_increment , `name` VARCHAR(255), `address` VARCHAR(255), `location` VARCH
AR(255), `phone` VARCHAR(255), `open_time` VARCHAR(255), `cell_phone` VARC
HAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, PR
IMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `Shops`
Executing (default): CREATE TABLE IF NOT EXISTS `ShopsMenu` (`id` INTEGER
auto_increment , `name` VARCHAR(255), `price` INTEGER, `createdAt` DATETIM
E NOT NULL, `updatedAt` DATETIME NOT NULL, `shop_id` INTEGER, PRIMARY KEY
(`id`), FOREIGN KEY (`shop_id`) REFERENCES `Shops` (`id`) ON DELETE CASCAD
E ON UPDATE CASCADE) ENGINE=InnoDB;
```

POST /admin/shops/detail/:id 저장되는 부분 작성

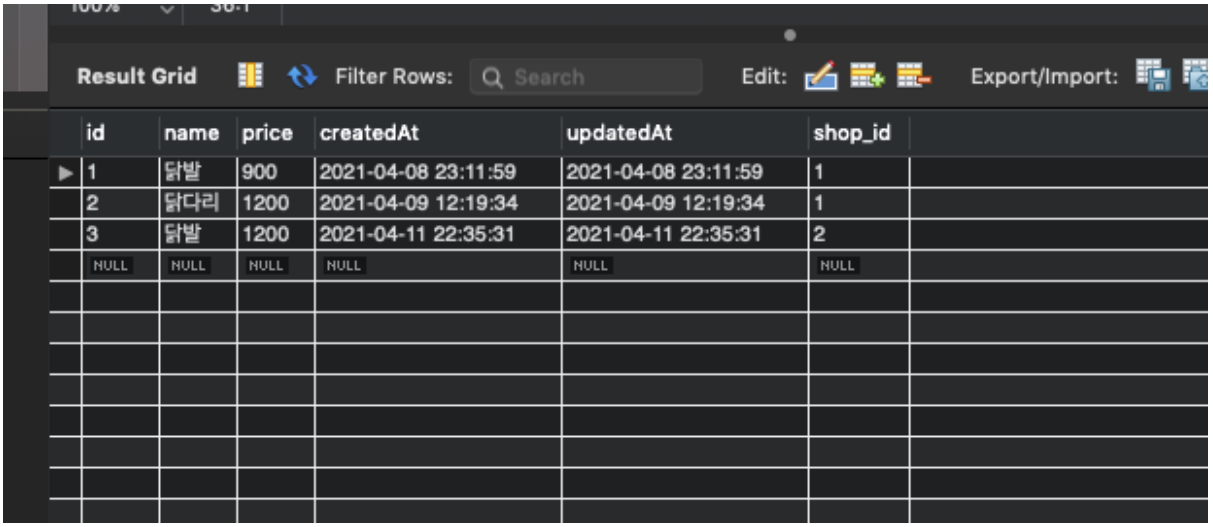
controllers/admin/index.js

```
router.get('/shops/detail/:id', ctrl.add_menu );
```

controllers/admin/admin.ctrl.js

```
exports.add_menu = async(req, res) => {

  try{

        const shop = await models.Shops.findByPk(req.params.id);
        // create + as에 적은 내용 ( shops.js association 에서 적은 내용 )
        await shop.createMenu(req.body);
        res.redirect('/admin/shops/detail/'+req.params.id);

  }catch(e){
        console.log(e)
  }

}
```

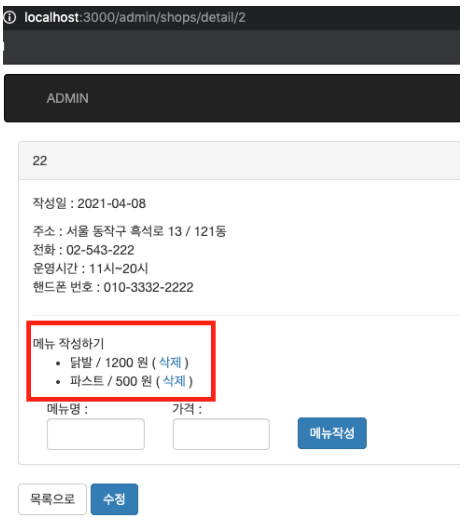
메뉴 저장해보고 workbench 에서 확인



id	name	price	createdAt	updatedAt	shop_id
1	닭발	900	2021-04-08 23:11:59	2021-04-08 23:11:59	1
2	닭다리	1200	2021-04-09 12:19:34	2021-04-09 12:19:34	1
3	닭발	1200	2021-04-11 22:35:31	2021-04-11 22:35:31	2
NULL	NULL	NULL	NULL	NULL	NULL

아래와 같이 메뉴 명을 뿌려지게 작성

GET /admin/shops/detail/:id



template/admin/detail.html

메뉴 작성하기

```
<ul>
  {% for menu in shop.Menu %}
    <li>
      {{ menu.name }} / {{ menu.price }} 원
      (
        <a href="/admin/shops/delete/{{ shop.id }}/{{ menu.id }}"
           onclick="return confirm('삭제하시겠습니까?')">
           삭제
        </a>
      )
    </li>
  {% endfor %}
</ul>
```

controllers/admin/admin.ctrl.js

아래와 같이 get_shops_detail 에 shop 데이터 수정

```
exports.get_shops_detail = async(req, res) => {

  try{

    const shop = await models.Shops.findOne({
       where : {
          id : req.params.id
       },
       include :[
          'Menu'
       ]
    });

    res.render('admin/detail.html', { shop });

  }catch(e){
     console.log(e)
  }

}
```

```
exports.get_shops_detail = async(req, res) => {

    try{

      const shop = await models.Shops.findOne({
          where : {
              id : req.params.id
          },
          include :[
              'Menu'
          ]
      });

      res.render('admin/detail.html', { shop });

    }catch(e){
        console.log(e)
    }


}
```

삭제하기 작성
controllers/admin/index.js

```
// 메뉴 삭제
router.get('/shops/delete/:shop_id/:menu_id', ctrl.remove_menu );
```

controllers/admin/admin.ctrl.js
하단 추가

```
exports.remove_menu = async(req, res) => {

 try{

   await models.ShopsMenu.destroy({
      where: {
         id: req.params.menu_id
      }
   });

   res.redirect('/admin/shops/detail/' + req.params.shop_id );

 }catch(e){

 }

}
```

삭제 눌러보기