Support

gestion simplifiée de l’inventaire des types d’équipement des salles d’un établissement universitaire

1. **Ecriture des fichiers « ts/js »**

Les fichiers agissant directement sur l’interface graphique seront sauvegardés dans le dossier « src/controleur » de l’application.

Les fichiers d’accès aux données seront sauvegardés dans le dossier « src/modele » de l’application.

Le travail d’écriture des fichiers du « contrôleur » peut et doit se faire conjointement avec l’écriture des fichiers du « modele ».

A des fins pédagogiques, on va écrire

* dans un premier temps les fichiers « ts/js »d’accès aux données de la base « bdinventaire » dans le dossier « modele »,
* puis dans un second temps les autres fichiers « ts/js » dans le dossier « controleur »

## Les fichiers « ts/js » d’accès aux données de la base

Rappel du schéma relationnel **voir présentation sujet ②**

DEPARTEMENT ( **code\_dept**, nom\_dept, resp\_dept )

SALLE ( **num\_salle**, lib\_salle, etage, code\_dept ) tel que code\_dept : clé étrangère

TYPE\_EQUIPT ( **id\_equipt**, lib\_equipt, commentaire )

contient ( **num\_salle**, **id\_equipt,** nb ) tels que num\_salle : clé étrangère ; id\_equipt : clé étrangère

Remarque

* « **SALLE** » est une relation **maître**,
* « **DEPARTEMENT** » est une relation **enfant** de SALLE
* « **TYPE\_EQUIPT** » est une relation **détail** liée par « contient » à « SALLE »

Dans le dossier « modele » de l’application, on va créer les fichiers

* connexion.ts contenant la commande de connexion à la base
* data\_departement.ts, data\_salle.ts, data\_equipement.ts contenant les classes métiers et les classes d’accès aux données avec les requêtes SQL, nécessaires au développement du travail demandé.

1er paramètre : dossier où se trouve votre fichier “inventaire.html”

2ème paramètre : dossier où se trouve le fichier “php” permettant l’exécution de vos requêtes d’accès à la base de données

**./src/modele/connexion.ts**

import \* as APIsql from "../modele/sqlWeb.js"

APIsql.sqlWeb.init("https://devweb.iutmetz.univ-lorraine.fr/~toto3u/ihm/tp\_inventaire/vue/"  
 ,"https://devweb.iutmetz.univ-lorraine.fr/~nitschke5/ihm/IHM\_API/")

class Connexion {

constructor() {

this.init();

méthode utilisée définies dans "sqlWeb.ts" du dossier "src/controleur"

**bdOpen**(host : string, port : string, bdname : string, user : string, pwd : string, charset ='utf8', driver ='mysql')

}

init():void {

// à adapter avec voter nom de base et vos identifiants de connexion

APIsql.sqlWeb.bdOpen('devbdd.iutmetz.univ-lorraine.fr','3306','toto3u\_bdinventaire'   
 , 'toto3u\_appli','motdepasse', 'utf8');

}

}

let connexion = new Connexion;

export {connexion, APIsql}

**./src/modele/data\_departement.ts** - DEPARTEMENT relation enfant de SALLE

import {connexion, APIsql} from "../modele/connexion.js"

class UnDept { // définition de la classe gérant les données d’un département

private \_codeDept : string;

private \_nomDept : string;

private \_respDept : string;

constructor(code\_dept = "", nom\_dept ="" , resp\_dept  = "" ) {  
 // initialisation à l’instanciation

this.\_codeDept = code\_dept;

this.\_nomDept = nom\_dept;

this.\_respDept = resp\_dept;

}

// définition des « getters » et des « setters » pour la lecture/écriture des attributs privés de la classe

get codeDept():string { return this.\_codeDept; }

set codeDept ( code\_dept : string ) { this.\_codeDept = code\_dept; }

get nomDept():string { return this.\_nomDept; }

set nomDept ( nom\_dept : string ) { this.\_nomDept = nom\_dept; }

get respDept():string { return this.\_respDept; }

set respDept ( resp\_dept : string ) { this.\_respDept = resp\_dept; }

toArray():APIsql.TtabAsso { // renvoie l’objet sous la forme d’un tableau associatif   
 // pour un affichage dans une ligne d’un tableau HTML

let tableau :  APIsql.TtabAsso = {'codeDept':this.codeDept, 'nomDept':this.nomDept, 'respDept':this.respDept };

return tableau;

}

}

type TDepts = {[key: string]: UnDept}; // tableau d’objets UnDept

// eslint-disable-next-line @typescript-eslint/no-unused-vars

class LesDepts { // définition de la classe gérant les données de la table DEPARTEMENT

constructor () {

types utilisés définis dans le fichier "sqlWeb.ts"

**type TtabAsso =** {   *// tableau "associatif" : clé/indice du tableau est une chaîne de caractères*

    [key:string] : string; }

**type TdataSet  = TtabAsso[];***// tableau de tableau associatif*

// rien

}

private load(result : APIsql.TdataSet) : TDepts {

// à partir d’un TdataSet, conversion en tableau d’objets UnDept

const depts : TDepts = {};

for (let i=0; i<result.length; i++) {

const item:APIsql.TtabAsso = result[i];

const dept = new UnDept(item['code\_dept'], item['nom\_dept'], item['resp\_dept']);

depts[dept.codeDept] = dept; // clé d’un élément du tableau : code dépt

}

return depts;

}

private prepare(where:string):string { // préparation de la requête avec ou sans restriction (WHERE)

let sql : string;

sql = "SELECT code\_dept, nom\_dept, resp\_dept  FROM  DEPARTEMENT ";

if (where !== "")

{

sql += " WHERE " +where;

}

return sql;

méthode utilisée de "sqlWeb.ts"

**SQLloadData**(sp : string, params : string[], req ='interrogation'):TdataSet

}

all() : TDepts { // renvoie le tableau d’objets contenant tous les départements

return this.load(APIsql.sqlWeb.SQLloadData(this.prepare(""),[]));

}

byCodeDept(code\_dept : string) : UnDept { // renvoie l’objet correspondant au département code\_dept

let dept = new UnDept;

const depts : TDepts =  
 this.load(APIsql.sqlWeb.SQLloadData(this.prepare("code\_dept = ?"),[code\_dept]));

const lesCles: string[] = Object.keys(depts);  
 // affecte les clés du tableau associatif « depts » dans le tableau de chaines « lesCles »

if ( lesCles.length > 0) {

dept = depts[lesCles[0]]; // récupérer le 1er élément du tableau associatif « depts »

}

return dept;

}

toArray(depts : TDepts) : APIsql.TdataSet {  
 // renvoie le tableau d’objets sous la forme d’un tableau de tableaux associatifs   
 // pour un affichage dans un tableau HTML

let T:APIsql.TdataSet = [];

for (let id in depts) {

T.push(depts[id].toArray());

}

return T;

}

}

export {connexion}

export {UnDept}

export {LesDepts}

export {TDepts}

transpiler et corriger les éventuelles erreurs

**./src/modele/data\_salle.ts** - SALLE relation maître

import {connexion, APIsql} from "../modele/connexion.js"

class UneSalle {

private \_numSalle : string;

private \_libSalle : string;

private \_etage : string;

private \_codeDept   : string;

constructor(num\_salle = "", lib\_salle ="" , etage ="", code\_dept = "") {

this.\_numSalle = num\_salle;

this.\_libSalle = lib\_salle;

this.\_etage = etage;

this.\_codeDept = code\_dept;

}

// définition des « getters » et des « setters » pour les attributs privés de la classe

get numSalle() : string { return this.\_numSalle; }

set numSalle(num\_salle:string) { this.\_numSalle = num\_salle; }

get libSalle(): string { return this.\_libSalle; }

set libSalle(lib\_salle:string) { this.\_libSalle = lib\_salle; }

get etage(): string { return this.\_etage; }

set etage(etage:string) { this.\_etage = etage; }

get codeDept(): string { return this.\_codeDept; }

set codeDept(code\_dept:string) { this.\_codeDept = code\_dept; }

toArray():APIsql.TtabAsso {

// renvoie l’objet sous la forme d’un tableau associatif   
 // pour un affichage dans une ligne d’un tableau HTML

const tableau : APIsql.TtabAsso = {'numSalle':this.\_numSalle, 'libSalle':this.\_libSalle  
 , 'etage':this.etage, 'codeDept':this.\_codeDept };

return tableau;

}

}

Rappel du travail demandé : gestion de la table « SALLE ». Dans la classe LesSalles, il faut écrire

* les méthodes qui permettent de gérer les données d’un enregistrement de « SALLE » :   
  existence, lecture, ajout, modification, suppression

type TSalles = {[key: string]: UneSalle }; // tableau d’objets UneSalle

// eslint-disable-next-line @typescript-eslint/no-unused-vars

class LesSalles { // définition de la classe gérant les données de la table SALLE

constructor () { // rien

}

idExiste(num\_salle : string) : boolean {   
 // renvoie le test d’existence d’une salle dans la table  
 // sert pour l’ajout d’une nouvelle salle

return (APIsql.sqlWeb.SQLloadData("SELECT num\_salle FROM SALLE WHERE num\_salle=?"  
 ,[num\_salle]).length > 0);

}

private load(result : APIsql.TdataSet) : TSalles {  
 // à partir d’un TdataSet, conversion en tableau d’objets UneSalle

let salles : TSalles = {};

for (let i=0; i<result.length; i++) {

const item:APIsql.TtabAsso = result[i];

const salle = new UneSalle(item['num\_salle'], item['lib\_salle'], item['etage'], item['code\_dept']);

salles[salle.numSalle] = salle; // clé d’un élément du tableau : num salle

}

return salles;

}

private prepare(where:string):string { // préparation de la requête avec ou sans restriction (WHERE)

let sql : string;

sql = "SELECT num\_salle, lib\_salle, etage, code\_dept ";

sql += " FROM SALLE";

if (where !== "")

{

sql += " WHERE " +where;

}

return sql;

}

all() : TSalles { // renvoie le tableau d’objets contenant toutes les salles

return this.load(APIsql.sqlWeb.SQLloadData(this.prepare(""),[]));

}

byNumSalle (num\_salle : string) : UneSalle { // renvoie l’objet correspondant à la salle num\_salle

let salle = new UneSalle;

const salles : TSalles = this.load(APIsql.sqlWeb.SQLloadData(this.prepare("num\_salle = ?")  
 ,[num\_salle]));

const lesCles: string[] = Object.keys(salles);  
 // affecte les clés du tableau associatif « salles » dans le tableau de chaines « lesCles »

if ( lesCles.length > 0) {

salle = salles[lesCles[0]]; // récupérer le 1er élément du tableau associatif « salles »

}

return salle;

}

toArray(salles : TSalles) : APIsql.TdataSet { // renvoie le tableau d’objets sous la forme   
 // d’un tableau de tableaux associatifs pour un affichage dans un tableau HTML

let T:APIsql.TdataSet = [];

for (let id in salles) {

T.push(salles[id].toArray());

}

return T;

}

méthode utilisée de "sqlWeb.ts"

**SQLexec**(sp : string, params : string[]) : Boolean

// pour les requêtes qui ne renvoient pas de résultat

delete(num\_salle : string):boolean { // requête de suppression d’une salle dans la table

let sql : string;

sql = "DELETE FROM SALLE WHERE num\_salle = ?";

return APIsql.sqlWeb.SQLexec(sql,[num\_salle]); // requête de manipulation : utiliser SQLexec

}

insert(salle : UneSalle):boolean { // requête d’ajout d’une salle dans la table

let sql : string; // requête de manipulation : utiliser SQLexec

sql = "INSERT INTO SALLE(num\_salle, lib\_salle, etage, code\_dept ) VALUES (?, ?, ?, ?)";

return APIsql.sqlWeb.SQLexec(sql,[salle.numSalle, salle.libSalle, salle.etage, salle.codeDept]);

}

update(salle : UneSalle):boolean { // requête de modification d’une salle dans la table

let sql : string;

sql = "UPDATE SALLE SET lib\_salle = ?, etage = ?, code\_dept = ? ";

sql += " WHERE num\_salle = ?"; // requête de manipulation : utiliser SQLexec

return APIsql.sqlWeb.SQLexec(sql,[salle.libSalle, salle.etage, salle.codeDept, salle.numSalle]);

}

}

export {connexion}

export {UneSalle}

export {LesSalles}

export {TSalles}

transpiler et corriger les éventuelles erreurs

gestion de la relation « TYPE\_EQUIPT » et de « contient »

**./src/modele/data\_equipement.ts**

import {connexion, APIsql} from "../modele/connexion.js"

class UnTypEquipt {

private \_idEquipt: string;

private \_libEquipt : string;

private \_commentaire : string;

constructor(id\_equipt = "", lib\_equipt ="" , commentaire ="")   
 { // initialisation à l’instanciation

this.\_idEquipt= id\_equipt;

this.\_libEquipt = lib\_equipt;

this.\_commentaire = commentaire;

}

// définition des « getters » et des « setters » pour les attributs privés de la classe

get idEquipt() : string  { return this.\_idEquipt; }

set idEquipt(id\_equipt:string)  { this.\_idEquipt = id\_equipt; }

get libEquipt() : string  { return this.\_libEquipt; }

set libEquipt(lib\_equipt:string)  { this.\_libEquipt = lib\_equipt; }

get commentaire() :string   { return this.\_commentaire; }

set commentaire(commentaire:string)  { this.\_commentaire = commentaire; }

toArray():APIsql.TtabAsso

{

// renvoie l’objet sous la forme d’un tableau associatif   
 // pour un affichage dans une ligne d’un tableau HTML

let tableau : APIsql.TtabAsso = {'idEquipt':this.\_idEquipt, 'libEquipt':this.\_libEquipt  
 , 'commentaire':this.\_commentaire};

return tableau;

}

}

type TTypEquipts = {[key: string]: UnTypEquipt }; // tableau d’objets UnTypEquipt

// eslint-disable-next-line @typescript-eslint/no-unused-vars

class LesTypEquipts { // définition de la classe gérant les données de la table TYPE\_EQUIPT

constructor () {

// rien

}

private load(result : APIsql.TdataSet) : TTypEquipts {

// à partir d’un TdataSet, conversion en tableau d’objets UnTypEquipt

let typEquipts : TTypEquipts = {};

for (let i=0; i<result.length; i++) {

const item:APIsql.TtabAsso = result[i];

const typEquipt = new UnTypEquipt(item['id\_equipt'], item['lib\_equipt'],  item['commentaire']);

typEquipts[typEquipt.idEquipt] = typEquipt; // clé d’un élément du tableau : id equipt

}

return typEquipts;

}

private prepare(where:string):string { // préparation de la requête avec ou sans restriction (WHERE)

let sql : string;

sql = "SELECT id\_equipt, lib\_equipt, commentaire";

sql += " FROM TYPE\_EQUIPT"

if (where.trim() !== "")

{

sql += " WHERE " +where;

}

sql += " ORDER BY lib\_equipt ASC ";

return sql;

}

all() : TTypEquipts { // renvoie le tableau d’objets contenant tous les équipements

return this.load(APIsql.sqlWeb.SQLloadData(this.prepare(""),[]));

}

byIdEquipt(id\_equipt:string) : UnTypEquipt { // renvoie l’objet correspondant à l’équipement id\_equipt

let typEquipt = new UnTypEquipt;

const typEquipts : TTypEquipts = this.load(APIsql.sqlWeb.SQLloadData  
 (this.prepare("id\_equipt = ?"),[id\_equipt]));

const lesCles: string[] = Object.keys(typEquipts);

// affecte les clés du tableau associatif « typEquipts » dans le tableau de chaines « lesCles »

if ( lesCles.length > 0) {

typEquipt = typEquipts[lesCles[0]]; // récupérer le 1er élément du tableau associatif « typEquipts »

}

return typEquipt;

}

toArray(typEquipts : TTypEquipts) : APIsql.TdataSet { // renvoie le tableau d’objets sous la forme   
 // d’un tableau de tableaux associatifs pour un affichage dans un tableau HTML

let T:APIsql.TdataSet = [];

for (let id in typEquipts) {

T.push(typEquipts[id].toArray());

}

return T;

}

}

Classe représentant la relation « contient ».

Le nom de la classe sera composée des noms des relations détail – maître, pour notre cas « TypEquiptBySalle ».

Cela indique que l’accès aux données de la relation détail « TYPE\_EQUIPT » se fait par l’identifiant « num\_salle » de la relation maître « SALLE »

class UnTypEquiptBySalle {

private \_qte : number;

private \_unTypEquipt : UnTypEquipt;

constructor(unTypEquipt : UnTypEquipt = null, qte =0) {

// attributs de TYPE\_EQUIPT auxquelles on ajouter l’attribut « qte » de la relation « contient »

this.\_unTypEquipt = unTypEquipt;

this.\_qte = qte;

}

// définition des « getters » et des « setters » pour les attributs privés de la classe

get qte():number { return this.\_qte; }

set qte(qte :number) { this.\_qte = qte; }

get unTypEquipt():UnTypEquipt              {   return this.\_unTypEquipt;       }

   set unTypEquipt(unTypEquipt : UnTypEquipt)  {   this.\_unTypEquipt = unTypEquipt;    }

toArray():APIsql.TtabAsso {

// renvoie l’objet sous la forme d’un tableau associatif   
 // pour un affichage dans une ligne d’un tableau HTML

let tableau = this.unTypEquipt.toArray(); // appel de la méthode « toArray » de « UnTypEquipt »

tableau['qte'] = this.qte.toFixed(0);

return tableau;

}

}

type TTypEquiptsBySalle = {[key: string]: UnTypEquiptBySalle };

// eslint-disable-next-line @typescript-eslint/no-unused-vars

class LesTypEquiptsBySalle {

constructor () {

// rien

}

    private load(result : APIsql.TdataSet) :    TTypEquiptsBySalle  {

    // à partir d’un TdataSet, conversion en tableau d’objets UnTypEquiptBySalle

        const typEquiptsBySalle : TTypEquiptsBySalle = {};

        const lesTypEquipts = new LesTypEquipts();

        for (let i=0; i<result.length; i++) {

            const item:APIsql.TtabAsso = result[i];

            const unTypEquipt = lesTypEquipts.byIdEquipt(item['id\_equipt']);

            const typEquiptBySalle = new UnTypEquiptBySalle(unTypEquipt, parseInt(item['qte']));

            typEquiptsBySalle[typEquiptBySalle.unTypEquipt.idEquipt] = typEquiptBySalle;

        }

        return typEquiptsBySalle;

    }

    private prepare(where:string):string {

        let sql : string;

        sql = "SELECT id\_equipt, qte";

        sql += " FROM   contient";

        if (where.trim() !== "")

        {

            sql += " WHERE " +where;

        }

        return sql;

    }

byNumSalle(num\_salle : string) : TTypEquiptsBySalle {   
 // renvoie le tableau d’objets contenant tous les équipements de la salle num salle

return this.load(APIsql.sqlWeb.SQLloadData(this.prepare("num\_salle = ?"),[num\_salle]));

}

byNumSalleIdEquipt(num\_salle : string, id\_equipt : string ) : UnTypEquiptBySalle {  
 // renvoie l’objet de l’équipement id\_equipt contenu dans la salle num\_salle

let typEquiptBySalle = new UnTypEquiptBySalle;

let typEquiptsBySalle : TTypEquiptsBySalle = this.load(APIsql.sqlWeb.SQLloadData  
 (this.prepare("num\_salle = ? and id\_equipt = ?"),[num\_salle, id\_equipt]));

if ( !typEquiptsBySalle [0] === undefined) {

typEquiptBySalle = typEquiptsBySalle[0];

}

return typEquiptBySalle;

}

toArray(typEquipts : TTypEquiptsBySalle) : APIsql.TdataSet {

let T:APIsql.TdataSet = [];

for (let id in typEquipts) {

T.push(typEquipts[id].toArray());

delete T[T.length -1].commentaire; // pas besoin du commentaire pour l'affichage dans le tableau

}

return T;

}

getTotalNbEquipt(typEquipts : TTypEquiptsBySalle) : number{   
 // renvoie la quantité totale d’équipements d’une salle

let total = 0;

for (let id in typEquipts) {

total +=  typEquipts[id].qte;

}

return total;

}

delete(num\_salle : string):boolean { // requête de suppression des équipements d’une salle dans «contient»

let sql : string;

sql = "DELETE FROM contient WHERE num\_salle = ?";

return APIsql.sqlWeb.SQLexec(sql,[num\_salle]); // requête de manipulation : utiliser SQLexec

}

insert(num\_salle : string,  typEquipts : TTypEquiptsBySalle):boolean {  
 // requête d’ajout des équipements avec une quantité dans « contient » installé dans « num\_salle »

let sql : string;

let separateur = "";

sql = "INSERT INTO contient(num\_salle,id\_equipt, qte) VALUES ";

    for (let cle in typEquipts) {

      sql += separateur +"('" +num\_salle +"','" +typEquipts[cle].unTypEquipt.idEquipt +"','" +typEquipts[cle].qte +"')";

      separateur = ",";

      }

return APIsql.sqlWeb.SQLexec(sql,[]);

}

}

export {connexion}

export {UnTypEquipt}

export {LesTypEquipts}

export {TTypEquipts}

export {UnTypEquiptBySalle}

export {LesTypEquiptsBySalle}

export {TTypEquiptsBySalle}

transpiler et corriger les éventuelles erreurs