IHM – Mode opératoire du développement d'une application avec gestion d'une base de données

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 »

4.1. 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.

./src/modele/connexion.ts

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

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

APIsql.sqlWeb.init("http://devweb.iutmetz.univ-lorraine.fr/~toto3u/ihm/tp_inventaire/vue/"
                  ,"http://devweb.iutmetz.univ-lorraine.fr/~nitschke5/ihm/IHM_API/")

class Connexion {
   constructor() {
      this.init();
   }
   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}
```

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')

./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 () {
      // 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;
   }

   all() : TDepts { // renvoie le tableau d'objets contenant tous les départements
      return this.load(APIsql.sqlWeb.SQLloadData(this.prepare(""),[]));
   }
```

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*

méthode utilisée de "sqlWeb.ts"

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

```
   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 ="" ) {
   // 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