COMPTE RENDU

Outils de gestion écologique

Explications technique de la réalisation du logiciel

Date de la dernière mise à jour : 03 juillet 2020

Sommaire

1. Ecriture du code

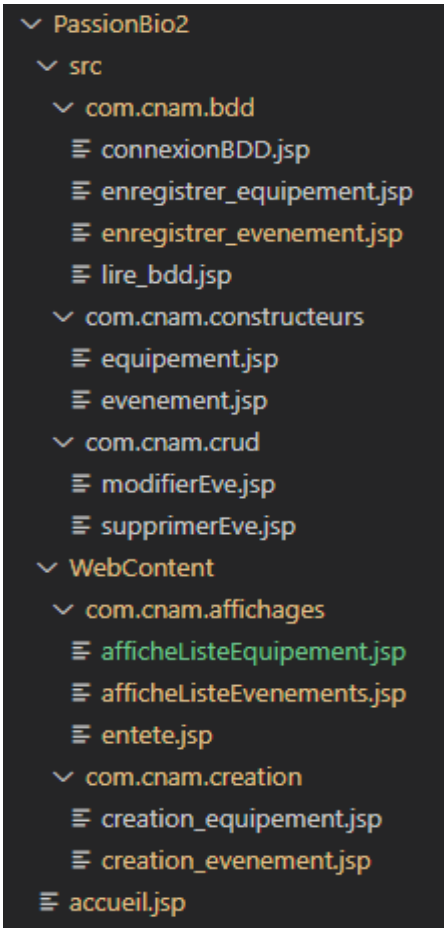
1.1 Choix du langage

Notre choix s'est orienté sur le Java car c'est le langage que nous "maîtrisons" le mieux.
Après de multiples recherches, nous avons constaté que la solution pour gérer à la fois des pages Web dynamiques et de la base de donnée SQL serait Java EE.

Suite à un abandon de cette solution que nous n'avons pas réussi à faire fonctionner (voir ci-dessous "Difficultés"), nous sommes partis sur la solution utilisée dans le cours NFA011, à savoir des pages JSP.

A MODIFIER : nous nous sommes aperçu que la méthode utilisée en NFA011 (JSP+SQL), est une solution en mode « bricolage ». Nous avons rencontré de nombreuses difficultés d'échanges entre les pages et notamment les liens relatifs qui ont un fonctionnement aléatoire (quand on fait un rafraichissement de page 9 fois sur 10 la page s'affiche correctement, sinon il y a un message d'erreur).

2 Structure du programme



PassionBio2 :La racine du projet (2 pour V2)

src : pages java
com.cnam.bdd : toutes les opérations sur la base de donnée.
- **connexionBDD.jsp** : connexion à la BDD
- **enregistrer_equipement.jsp** : requête SQL qui enregistre les équipements
- **enregistrer_evenement.jsp** : requête SQL qui enregistre les événements
- **lire_bdd.jsp** : lit les informations (événement + équipement)

com.cnam.contructeurs : pages des constructeurs
- **equipement.jsp** : Constructeur équipements
- **evenement.jsp** : Constructeur événements

com.cnam.crud : lecture et écriture des données dans la BDD
- **modifierEve.jsp** : modification des événements
- **supprimerEve.jsp** : suppression des événements

Webcontent : pages web
com.cnam.affichages : affichage des informations
- **afficheListeEquipement.jsp** : affiche sous forme de tableau les événements.
- **afficheListeEvenement.jsp** : affiche sous forme de tableau les équipements

com.cnam.creation : saisie des informations
- **creation_equipement.jsp** : formulaire de création d'équipement
- **creation_evenement.jsp** : formulaire de création d'événements.

accueil.jsp : page d'accueil

A MODIFIER : par manque de temps, nous n'avons pas pu réorganiser la structure.
il aurait fallu mettre :

- Enregistrer_equipement.jsp → dans com.cnam.crud.
- Enregistrer_evenement.jsp → dans com.cnam.crud
- Lire_bdd → dans com.cnam.crud

2.1 Exemples de codes du programme

2.1.1 Constructeur pour stocker les informations liées à un événement

```
final class Equipement {
    /*
     ******************  VARIABLES  ***********************
     NumEquip           Entier 11 chiffres          Clé primaire
     dispoEquip         Entier
     nomEquip           string 30 caracteres
     totalEquip         Entier 11
     dateEmprunt        date
     heureEmprunt       time
     numEmp             Entier 11

     *****************  CHAMPS DATES ***********************
     dateRetour         date
     heureRetour        time
     */

        private int NumEquip;
        private String nomEquip;
        private int dispoEquip;
        private int totalEquip;
        private String dateEmprunt;
        private String heureEmprunt;
        private String dateRetour;
        private String heureRetour;
        private int numEmp;



    public Equipement() {

      this.NumEquip=0;
      this.nomEquip="";
      this.dispoEquip=0;
      this.totalEquip=0;
      this.dateEmprunt="";
      this.heureEmprunt="";
      this.dateRetour="";
      this.heureRetour="";
      this.numEmp=0;
```

```
    // constructeur avec parametres
     public Equipement(String nomEquip, int dispoEquip, int totalEquip, String date
Emprunt, String heureEmprunt, String dateRetour, String heureRetour) {

       this.NumEquip=0;
       this.nomEquip=nomEquip;
       this.dispoEquip=dispoEquip;
       this.totalEquip=totalEquip;
       this.dateEmprunt=dateEmprunt;
       this.heureEmprunt=heureEmprunt;
       this.dateRetour=dateRetour;
       this.heureRetour=heureRetour;
       this.numEmp=numEmp;

    }

    public int getNumEquip() {
        return NumEquip;
    }

    public void setNumEquip(int numEquip) {
        NumEquip = numEquip;
    }

    public String getNomEquip() {
        return nomEquip;
    }

    public void setNomEquip(String nomEquip) {
        this.nomEquip = nomEquip;
    }

    public int isDispoEquip() {
        return dispoEquip;
    }

    public void setDispoEquip(int dispoEquip) {
        this.dispoEquip = dispoEquip;
    }

    public int getTotalEquip() {
        return totalEquip;
    }

    public void setTotalEquip(int totalEquip) {
        this.totalEquip = totalEquip;
    }

    public String getDateEmprunt() {
```

2.1.2 Page d'affichage du formulaire création d'événements.

Cette page permet la saisie des informations de la création d'un événement.
Au clic sur le bouton « valider » le programme renvoi vers la page « enregistrer_evenement.jsp » qui enregistre ces données dans la BDD en SQL.

```
<%@page contentType="text/html" language="java" import="java.sql.*,java.text.*" pageEncoding="UTF
-8" errorPage="" %>
<%@page import="java.time.*"%>


<%@include file="../../WebContent/com.cnam.affichages/entete.jsp"%>   <!-- header-->
<div class="container" style="margin-top:30px">
       <!-- creation du formulaire de saisie -->
       <form class="form-
horizontal" action="/PassionBio2/src/com.cnam.bdd/enregistrer_equipement.jsp" method="post" name=
"ajout_equipement" name="add_equipement">

          <fieldset>

               <!-- Formulaire équipement -->
              <legend>Création d'équipement</legend>

              <!-- Input pour le  Nom-->
              <div class="form-group">
              <label class="col-md-4 control-label" for="nomEquip">Nom</label>
              <div class="col-md-4">
              <input id="nomEquip " name="nomEquip" type="text" placeholder="Nom de l'équipemen
t" class="form-control input-md" required="">

              </div>
              </div>

              <!--  Input pour la date d'emprûnt-->
              <div class="form-group">
              <label class="col-md-4 control-label" for="dateEmprunt">Date d'emprûnt</label>
              <div class="col-md-3">
              <input id="dateEmprunt" name="dateEmprunt" type="date" placeholder="Date d'emprun
t" class="form-control input-md" required="">
              </div>

               <!--  Input pour la date d'emprûnt-->
              <div class="form-group">
              <label class="col-md-4 control-
label" for="heureEmprunt">Heure d'emprunt</label>
              <div class="col-md-4">
              <input id="heureEmprunt" name="heureEmprunt" type="text" placeholder="Heure d'emp
runt" class="form-control input-md" required="">
              </div>

                <!--  Input pour la date de retour -->
              <div class="form-group">
              <label class="col-md-4 control-label" for="dateRetour">Date de retour</label>
              <div class="col-md-3">
              <input id="dateRetour" name="dateRetour" type="date" placeholder="Date de retour
" class="form-control input-md" required="">


                  <!--  Input pour la quantité emprûnté -->
                <div class="form-group">
                <label class="col-md-4 control-
label" for="nbrEmpruntEquip">Nombre d'emprûnt</label>
                <div class="col-md-2">
                <input id="nbrEmpruntEquip" name="nbrEmpruntEquip" type="number" placeh
older="Nombre" class="form-control input-md" required="">

                </div>



                <!-- Buttons -->
                <div class="form-group">
                <label class="col-md-4 control-
label" for="button_crea_equip_ok"></label>
                <div class="col-md-8">
                    <button id="button_crea_equip_ok" name="button_crea_equip_ok" class
="btn btn-success">Valider</button>         <!-- Validation du formulaire -->
                    <button id="button_crea_equip_reset" name="button_crea_equip_reset"
 class="btn btn-danger">Effacer</button>     <!-- Effacer le formulaire -->
                </div>

            </fieldset>
        </form>
        </div>



        </fieldset>
        </form>
    </div>


  </body>
  </html>
```

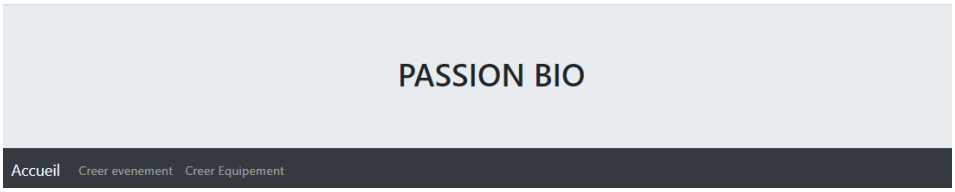
A AMELIORER : Validation des champs. Il faut vérifier que chaque champ soit conforme avant d'envoyer les informations en BDD, par exemple, vérifier que la date de réservation de soit pas antérieure à aujourd'hui car on ne peut pas réserver du matériel avant aujourd'hui.

2.1.3 Enregistrement du formulaire en BDD

```
<%@page contentType="text/html" language="java" import="java.sql.*,java.text.*" pageEncoding="UTF-
8" errorPage="" %>
<%@page import="java.time.*"%>
<%@page import="java.util.ArrayList"%>
<%@include file="./connexionBDD.jsp"%>
<%@include file="../com.cnam.constructeurs/equipement.jsp"%>
<%@include file="../com.cnam.constructeurs/evenement.jsp"%>


<%
    ArrayList<Evenement> arrayEvenement = new  ArrayList<Evenement>();

            // on recupère les informations saisies dans le formulaire
    String nom_event = request.getParameter("nom_event");
    String date_event = request.getParameter("date_event");
    String heure_event = request.getParameter("heure_event");
    int duree_event = Integer.parseInt(request.getParameter("duree_event"));
    int  nbPlace_event =  Integer.parseInt(request.getParameter("nbPlace_event"));
    int noteEvent=15;
    int nbrParticipant = 8;
    int numEmp =25;

    // Création d'un événement
        Evenement eve = new Evenement(nom_event,1,duree_event,nbPlace_event,15);
                // ajout de l'evenement dans l'arrayList 'arrayEvenement'
        arrayEvenement.add(eve);

                // Boucle For Each qui permet de vérifier si un événement du même nom est créé
        Statement instruction = conn.createStatement();
    try{

        for(Evenement elem: arrayEvenement){
                ResultSet resultat = instruction.executeQuery("SELECT COUNT(nomEve) nb_nomEve FROM evenement
WHERE nomEve='"+nom_event+"'");    //compte le nombre de champ 'nomEve'
                resultat.next();
                int nbEvent =(resultat.getInt("nb_nomEve"));        // nombre de champs identique 'nomEve' tr
ouvé
                out.println("<h2>BRAVO </h2>"+nbEvent);


           if (nbEvent<1){                                                  // si le nombre de 'nomEve' est i
nférieur à 1 c'est qu'il n'y en a pas de créé

                // Insertion des données du formulaire de saisie de création d'un événement dans la BDD
                int nb = instruction.executeUpdate("INSERT INTO evenement(dateEve,nomEve,noteEve,heureDebutEv
e,dureeEve,placeMaxEve,nbrParticipant,numEmp)"+
                "VALUES('"+date_event+"','"+nom_event+"','"+noteEvent+"','"+heure_event+"','"+duree_event+"',
'"+nbPlace_event+"','"+nbrParticipant+"','"+numEmp+"')");

                            // message pour dir que l'enregistrement est ok
                out.println("<h2>BRAVO l'enregistrement à été ajouté</h2>");
```

2.1.4 Page qui affiche le header

Ce fichier ’entete_jsp’ affiche le header qui apparait sur chaque page.
Il fait apparaitre le nom de la société ‘PassionBio’ et le menu de navigation.
Le but est que chaque page HTML du site fasse appel à cette page afin de ne pas répéter le code à chaque page et faciliter la modification. Si on veut ajouter un menu, la modification se fait une seule fois dans ’entete_jsp’.




```
<!DOCTYPE html>
<html lang="fr">
<head>
  <title>Bootstrap 4 Website Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
  <style>
  .fakeimg {
    height: 200px;
    background: #aaa;
  }
  </style>
</head>
<body>
<div class="jumbotron text-center" style="margin-bottom:0">
  <h1>PASSION BIO</h1>

</div>

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <a class="navbar-brand" href="../../accueil.jsp">Accueil</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#collapsibleNavbar">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="collapsibleNavbar">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="http://localhost:8081/PassionBio2/WebContent/com.cnam.creation/creation_evenement.jsp">Creer evenement</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="http://localhost:8081/PassionBio2/WebContent/com.cnam.creation/creation_equipement.jsp">Creer Equipement</a>
      </li>
    </ul>
  </div>
</nav>
```

2.1.4 Page qui affiche supprime un événement

Cette page supprime l'évènement sélectionné par l'utilisateur dans le liste des événements. Le programme récupère l'ID de l'événement quand l'utilisateur clique sur le bouton 'supprimer' puis grâce à une commande SQL supprime l'enregistrement

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.*"%>

<!-- L'UTILISATEUR NE VERA JAMAIS CETTE PAGE, CAR CELLE-
CI NE SERT QU'A EFFECTUER L'OPERATION DE SUPPRESSION -->

    <%
    try{
        // CHARGEMENT DU DRIVER
        String url = "jdbc:mysql://localhost:3308/bdpassionbio";
        String login = "root";
        String password = "";

        Class.forName("com.mysql.cj.jdbc.Driver");
        System.out.println("Connexion OK");
        Connection conn = DriverManager.getConnection(url, login, password);

        // CREATION DE L'OBJET STATEMENT
        Statement statement = conn.createStatement();
        // ON RECUPERE L'ID VENANT DU BOUTON DE LA PAGE ACCEUIL.JSP
        String id = request.getParameter("id_reference");
        // ON CONVERTIE L'ID DE TYPE STRING EN TYPE INT
        int numId = Integer.parseInt(id);

        // REQUETE SQL AFIN DE SUPPRIMER TOUTES LES INFOS D'UN EVENEMENT DE LA BDD
        String requete = " DELETE FROM evenement WHERE numEve = '" + numId + "'";
        // EXECUTION DE LA REQUETE
        statement.executeUpdate(requete);
        // ON RENVOIE L'UTILISATEUR DIRECTEMENT SUR LA PAGE D'ACCEUIL
        response.sendRedirect("../../accueil.jsp");

    }catch(Exception e) {   // MESSAGE D'ERREUR AU CAS OU IL Y AURAIT UN BUG

        out.println(request.getParameter("id_reference"));
    }

    %>
```

AMELIORATIONS : Mettre un pop-up pour demander si l'utilisateur veut vraiment supprimer l'enregistrement afin d'éviter une suppression par inadvertance.

Plutôt que de supprimer définitivement l'enregistrement, il aurait été plus judicieux d'ajouter un champs 'eveDel' qui passerai soit à 0 (événement supprimé) ou 1(événement actif). Ceci permettrait de garder un historique dans la base de données qui peut servir ultérieurement, comme par exemple pour des statistiques.

4. Difficultés

Java EE : La phase d'apprentissage a été un peu longue car il n'est pas évident de comprendre le fonctionnement.
Après avoir développé toute la partie de la gestion de connexion des utilisateurs, nous avons butés sur l'établissement de la connexion à la base de donnée. Même avec de l'assistance des professeurs, nous n'y sommes pas arrivés.
Refonte du projet : Nous avons dû abandonner cette voie pour partir sur une autre solution.

Git : Afin de partager et garder un historique de notre travail, nous avons choisi Git, mais nous avons eu des problèmes liés à l'utilisation de cet outil qui nécessite une formation. De ce fait, des commande git ont engendré des problème de versions.

5. Evolutions possibles

On pourrait ajouter une interface d'administration (permissions) qui permettrai de gérer les droits d'accès en création, modification, suppression d'événement et d'équipements.
Mais aussi la possibilité de créer des groupes afin de proposer des services uniquement à ceux-ci, comme par exemple, faire une événement uniquement avec la direction.

Développer une version pour les smartphone.
Avoir un bilan des bénéfices, sur le plan écologique, apporté par les animations et service apportés.
Personnalisation de l'interface avec des choix de couleurs ou de format de présentations.